Requirements elicitation through viewpoint control in a natural language environment

Mohammed Messaoudi

Requirements Elicitation Through Viewpoint Control in a Natural Language Environment

Mohammed Messaoudi

A thesis submitted for the degree of

Doctor of Philosophy

School of Engineering and Applied Science
(Computer Science)
University of Durham

1994

Abstract

While requirements engineering is about building a conceptual model of part of reality, requirements validation involves assessing the model for correctness, completeness, and consistency. Viewpoint resolution is the process of comparing different views of a given situation and reconciling different opinions. In his doctoral dissertation Leite [72] proposes viewpoint resolution as a means for early validation of requirements of large systems. Leite concentrates on the representation of two different views using a special language, and the identification of their syntactic differences. His method relies heavily on redundancy: two viewpoints (systems analysts) should consider the same topic, use the same vocabulary, and use the same rule-based language which constrains how the rules should be expressed. The quality of discrepancies that can be detected using his method depends on the quality of the viewpoints.

The hypothesis of this thesis is that, independently of the quality of the viewpoints, the number of viewpoints, the language, and the domain, it is possible to detect better quality discrepancies and to point out problems earlier than Leite's method allows. In the first part of this study, viewpoint-oriented requirements engineering methods are classified into categories based on the kind of multiplicity the methods address: multiple human agents, multiple specification processes, or multiple representation schemes. The classification provides a framework for the comparison and the evaluation of viewpoint-based methods. The study then focuses on the critical evaluation of Leite's method both analytically and experimentally. Counter examples were designed to identify the situations the method cannot handle.

The second part of the work concentrates on the development of a method for the very early validation of requirements that improves on Leite's method and pushes the boundaries of the validation process upstream towards fact-finding, and downstream towards conflicts resolution. The Viewpoint Control Method draws its principles from the fields of uncertainty management and natural language engineering. The basic principle of the method is that, in order to make sense of a domain one must learn about the information sources and create models of their behaviour. These models are used to assess pieces of information, in natural language, received from the sources and to resolve conflicts between them. The models are then reassessed in the light of feedback from the results of the process of information evaluation and conflict resolution. Among the implications of this approach is the very early detection of problems, and the treatment of conflict resolution as an explicit and an integral part of the requirements engineering process. The method is designed to operate within a large environment called LOLITA that supports relevant aspects of natural language engineering.

In the third part of the study the Viewpoint Control Method is applied and experimentally evaluated, using examples and practical case studies. Comparing the proposed approach to Leite's shows that the Viewpoint Control Method is of wider scope, is able to detect problems earlier, and is able to point out better quality problems. The conclusions of the investigation support the view that underlines the naivety of assuming competence or objectivity of each source of information.

To my uncle Mohamed.

In memory of my father.

Acknowledgements

This work has been sponsored by an Algerian government scholarship.

I am grateful to my supervisor Professor K.H. Bennett for his encouragements and guidance throughout this study. I am grateful to Mr. Malcolm Munro and Dr. Roberto Garigliano for their support.

This thesis has been produced using the LaTeX text formating system.

Contents

Chapter 1

Introduction

1.1 The Problem and its Context

In the process of software construction and before the technical process of the design can occur, one must understand the problem to be solved and have a clear picture of what is to be designed and built. That is the primary purpose of requirements engineering. According to *IEEE standard* [1] a requirement is:

> *(1) A condition or a capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document. The set of all requirements forms the basis for subsequent development of the system or system component.*

Specification denotes the document produced by the requirements engineering. Figure 1.1 shows the subprocesses of the requirements engineering process (according to Leite [72]). Requirements elicitation is the process of gathering and understanding requirements-related information. Elicitation

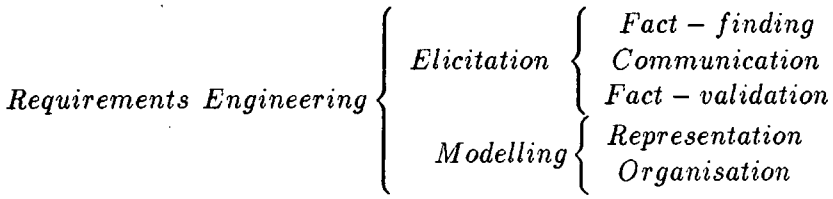

Figure 1.1: Requirements Engineering

involves fact-finding, validating one's understanding of the information gathered, and communicating open issues for negotiation. Fact-finding uses mechanisms such as interviews, questionnaires, and observation of the operational environment in which the proposed system will reside. Modelling involves creating a representation of the elicitation results in a form that can be analysed and reviewed with those who provided the information.

1.1.1 Uncertainty in Requirements Engineering

It is common for software engineering textbooks to stress the importance and the difficulties of requirements engineering, in general and requirements elicitation in particular [95]*.

Brooks [29], for example, summarises the "story" of requirements engineering as:

> *The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interfaces to other software systems. No part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later..*

Although there is no agreed definition for requirements engineering there is a consensus that part of the 'requirements stage' (regardless of the definition) is to understand the problem and to communicate that understanding, in the form of a representation, to others [103, 60, 72, 36].

*'One of the Seven (Plus or Minus Two) challenges for requirements research'

Problems, in turn, need to be **investigated** and explored before they can be understood. London [76] summarised the objectives of an investigation as:

> *To collect the maximum of correct, relevant information in the minimum of time, whilst performing the necessary 'public relations' functions.*

The early stages of requirements elicitation are characterised by often complex learning and knowledge acquisition tasks. The aim is to shape a clear picture of the problem for the analyst and for the customer. The more efficient the learning process, the more productive is requirements elicitation [70].

As indicated by Brooks, the elimination of errors at the earliest moment is a key to improving the productivity of the development process.

While a cause of the problems with software projects is the misunderstanding of the requirements a major cause of the misunderstanding is the **uncertainty** and fuzziness that characterise the early stages of the software development process [38, 71, 45]. The following are the major sources of uncertainty:

- Complexity of systems. As the technology advances, society gets more sophisticated and needs more complex systems with complex requirements.
- The informal nature of the changing environment (e.g., the organisational setting and the business milieu), in which the system will reside, causes some degree of uncertainty in the requirements [88, 45].
- Uncertainty about what is relevant information for collection. Requirements are difficult to uncover, especially for complex systems in unfamiliar domains, regardless of the techniques used.
- The natural language information. Although almost all research in requirements engineering today deals with the introduction of formal methods into the process of requirements

engineering, in practice most projects use natural language for expressing the requirements, partly because it is preferred by users and enhances communication. So there is a recognition of the need for more research to provide analysts with guiding principles to reduce the ambiguity inherent in information expressed in natural language. Natural language processing is an area in its own right. An existing natural language engineering system able to offer solutions to the issues related to the treatment of natural language information can be used as an environment for requirements engineering methods. It it the case for this work.

- The information sources. Information sources typically tend to be reliable only up to a point and differ in their reliability. As instruments tend to have a margin of error which affects the data they produce, human sources, whether single or compound, differ in their behaviour, thus making it difficult to gauge how reliable the information received from them is. Unlike any other source of information, however, people are capable of manipulating the information before passing it on: be that by added reasoning, influence of beliefs, or sheer deceit. Davis [38] identifies the user's degree of commitment, their experience and their area of responsibility as important factors in reducing or increasing the uncertainty.

- The difficulties in communication between the parties involved in requirements engineering [59]. Requirements engineering is communication-intensive. Because users and developers have different languages, have different professional background and tastes, the actual message may get distorted, thereby adding an element of uncertainty to the information exchanged.

Lehman [71] suggests that uncertainty is a direct consequence of nature, and that software engineering is really an attempt to manage that uncertainty.

Consider the following account [76]:

> *System analyst A: On my last project we were looking at the computerization of the accounts payable system. The project schedule was ridiculous: we had to cut back on the investigation. This meant we had to rely on a description of the existing system from one or two key users, mostly the Accounts Receivable Manager. I spent three days with this man. His explanation was a dream: clear, concise, and comprehensive. We were*

well into programming when I was browsing through some papers that came to light in an office move. I found a report that was seven years old. It had been written by the Accounts Receivable Manager when he was an ordinary clerk, describing a proposed new manual system. It was word-for-word the same as the description of the system he had given to me. But there was a stack of correspondence attached which turned the system down; it was never implemented. So there we were, basing a new system on a rejected idea that was seven years old. He had been pushing for this new system all these years and used the computer to get it. If we had gone on with the system, it would have been a disaster. We scrapped it and started again.

The principle that more sources of information provide a better understanding of a subject than a global source has triggered many research projects that adopt the idea of **multiple viewpoints** with respect to software construction, especially with respect to requirements engineering. However, the existing, viewpoint-oriented methods have concentrated on the handling the complexity that the use of viewpoints allows. The use of viewpoints allows the separation of concerns, i.e., details about one viewpoint can be ignored while developing another viewpoint; the distributed development, i.e., several people may independently develop different viewpoints, and merge them thereafter; and the explicit combination of different viewpoints, i.e., the explicit handling of conflicts between viewpoints. But these methods have ignored the role of controlling the information sources considered as a potential source of uncertainty. As pointed out by Lam [70], these methods have ignored the initial 'fuzzy' nature of the systems and seem to assume a more 'stable' problem domain. Following Brooks, uncertainty is an essential difficulty in requirements engineering rather than an accidental one. The uncertainty should not, therefore, be ignored.

1.2 The Validation Problem

The software artifact to be built is a component of a larger system that comprises hardware, people, procedures and other software systems. Only within the context of the entire system, and the interactions among its parts, can the behaviour of the proposed software system be defined [17]. The requirements engineering process is highly dependent upon the system engineering (or context analysis [104]) that sets the context and the constraints under which the proposed software system

will be developed [28]. Context analysis for a spreadsheet package, for example, will typically produce the results of a market analysis and a list of important product features. The product of a context analysis is called **universe of discourse**. The universe of discourse includes all the sources of information and all the people affected by the software. These people are referred to as actors in the universe of discourse [74].

Requirements validation is part of requirements elicitation and is responsible for ensuring that the requirements match the stake holders' intent. Validation is usually defined via the following question [22]:

`Are we building the right product?`

Since there is no formal mapping between the acquired facts and the original intent, software engineering research has been concentrating on improving the approximation between the requirements and the universe of discourse. A validation method should:

- detect wrong information, inconsistencies, and missing information, with respect to the universe of discourse as early as possible,
- allow for traceability between the information and the universe of discourse,
- encourage the users' involvement in the process,
- support the negotiation process for resolving the problems with the requirements,
- deal with changes in the requirements.

The principle of this work is that problems with requirements can be detected at an earlier stage in the requirements engineering process than existing methods allow. The issue addressed in this thesis can be illustrated by the following example:

> An engineering configuration manager decided to purchase a word processor for his secretary claiming that it will improve the quality of the control construction. From the secretary's point of view the word processor will save time. The item's cost was, at

the time, estimated at 7,000 pounds - a two-year salary for a secretary. The problem for the financial department was to establish whether the manager has a business case or merely wants a new toy and prestige for his secretary.

1.3 Research Method and Objectives

A *testing-out* research method has been pursued in this thesis [93, 66]. Fundamentally, a testing-out research method consists of finding the limits of previous generalisations and developing solutions to overcome those limits. Leite's method was used as a vehicle for the investigation. Leite is the only researcher that uses viewpoint resolution as a means for early validation of requirements of large systems. Leite concentrates on the representation of two different views using a special language, and the identification of their syntactic differences. His method relies heavily on redundancy: two viewpoints (systems analysts) should consider the same topic, use the same vocabulary, and use the same rule-based language which constrains how the rules should be expressed. The quality of discrepancies that can be detected using his method depends on the quality of the viewpoints.

The hypothesis of this thesis is that, independently of the quality of the viewpoints, the number of viewpoints, the language, the domain, it is possible to detect better quality discrepancies and to point out problems earlier than Leite's method allows.

Firstly, Leite's method is critically evaluated both analytically and experimentally. Counter examples were designed to identify the situations the method cannot handle. Secondly, an AI model called the Source Control Mechanism (SCM) is adapted to the domain of requirements engineering from multiple viewpoints. The SCM models how humans deal with the management of uncertainty with respect to natural language information received from multiple human sources. Thirdly, a large natural language engineering system called LOLITA (Large-scale Object-based Linguistic Interactor, Translator and Analyser) is used as an environment for the method developed in this thesis. LOLITA is used as a tool that offers solutions to the issues related to the treatment of natural language information.

The main objective of this work is to develop a validation method that has the characteristics set in the previous section. The aims are to:

1. detect wrong information, inconsistencies, and missing information, with respect to the universe of discourse as early as possible,
2. allow for traceability between the information and the universe of discourse,
3. encourage the users' involvement in the process,
4. support the negotiation process for resolving the problems with the requirements,
5. deal with changes in the requirements.

1.4 The Criteria for Success

The differences between validation methods are the type (i.e. inconsistencies, incompleteness, and incorrectness) and the quality of the problems, that can be detected, and the level of support that can be provided to the conflict resolution process. The set of problems detected and information about the 'roots' of those problems is called an agenda which is the input to the negotiation process. The quality of such an agenda is qualified by the level of support it provides the negotiation process.

The success of the developed method should be assessed in the context of the existing, early validation methods, particularly in the context of the Leite method (the most successful method in this area). The criteria for success are:

1. the ability to detect problems earlier than the Leite method allows,
2. the ability to deal with incorrectness,
3. the ability to provide a better quality agenda,
4. the ability to deal with conflict resolution.

The success of the described method will be assessed in Chapter 9.

1.5 Author's Contribution

The contribution of this work to software engineering research is a method for the very early validation of requirements using multiple viewpoints. The Viewpoint Control Method represents a novel approach to requirements elicitation. The principles of the method are drawn from the fields of uncertainty management, viewpoint resolution, and natural language engineering. The way in which these fileds are drawn together is novel, as is their application in the area of requirements engineering. In addition, the method has the following aspects:

1. The maintenance of **viewpoint models** and their use for assessing information from a **natural language** input.
2. The association of each participant with the information they contributed, and the recording of the analysis processes and their results, thus allowing the replay of those processes.
3. The treatment of conflict resolution as an explicit, and an integral part of the validation process.
4. The validation of natural language information.
5. The explicit use of human factors and relations in requirement engineering.

1.6 Overview of the Thesis

Viewpoint-based methods are not well known in the software engineering community. Chapter 2 introduces the software engineering context of the work and develops a new classification scheme for viewpoint-based methods. Most of the existing methods concentrate on requirements modeling,

ignoring requirements elicitation. Viewpoint Analysis (Leite's)is one of these methods. Chapter 3 reviews the viewpoint analysis method and provides a critical evaluation both analytically and experimentally.

Chapter 4 argues the case for real Natural Language Processing systems as a tool for requirements engineering. Various approaches are discussed, and the fundamental common drawback is pointed out: these may manifest themselves in the need for a pseudo-language, or heavy domain dependency, or excessive reliance on user interaction, but the drawbacks are due to the lack of proper NLP facilities (at the analysis, reasoning, and generation stages). The Natural Language Engineering paradigm has been briefly presented, and a claim made that is the way forward for real NLP. The LOLITA system is presented, in relation to this paradigm, and shown to have the range of functionality needed for many aspects of requirements engineering, requirements elicitation in particular.

Chapter 5 describes a set of principles for a new approach to viewpoint resolution that stresses the role of uncertainty in the information acquisition process and the crucial role that human factors and relations play in dealing with the uncertainty. The viewpoint resolution principles are the result of adapting the Source Control Mechanism to the area of requirements engineering. The Source Control Mechanism is introduced as a system for the mangement of uncertainty. The adaptation of the SCM is then detailed.

Chapter 6 describes a method for the very early validation of requirements based on the viewpoint resolution principles, introduced in Chapter 5. Chapter 7 describes the application of the method to an example and two major case studies to illustrate the analysis techniques of the method and the conflict resolution strategy.

Chapter 8 discusses the worth of the Viewpoint Control method as an early validation technique. The strengths and weaknesses of the method are discussed and the method is compared and contrasted with other methods. Chapter 9 provides a summary of the investigation. The objectives which have been achieved are discussed before the effect of the results of the work on the re-

quirements engineering process and the wider field. Finally, suggestions for further research are given.

A summary is given at the end of each chapter.

Chapter 2

Viewpoints in Requirements Engineering

Viewpoint-oriented software engineering is an emerging area of research. This chapter establishes the software engineering context for viewpoint-based requirements engineering and then gives a classification of the existing methods. A viewpoint method is seen here as a requirements engineering process of identifying viewpoints, reasoning within a viewpoint, reasoning between different viewpoints, and revising a viewpoint. The chapter ends with a summary of the common issues encountered by the methods.

2.1 Introduction

In many fields, it has been found necessary to take account of many ways of looking at some subject matter. Multiplicity appears in various guises in software engineering: view integration in software development environments [81], the multi-paradigm development [117] and N-version programming

[34]. Multiplicity also appears in other areas such as data base design [19] distributed artificial intelligence [110], belief systems, and distributed problem-solving. Recently many research groups have being addressing the idea of multiple viewpoints with respect to requirements engineering. Mullery declares that:

> *...The difficulties are often compounded by failure to recognise that what is needed is not one, but several expressions of requirements. The requirements expression must recognise several views of the system. Major aims must be: separation of different viewpoints, consistency and compatibility of the information in the overlap between viewpoints, and avoidance of unnecessary repetition in producing information common to more than one viewpoint.*
>
> [87, page12]

2.2 Software Life Cycle

A number of paradigms for the development of software have been proposed. The waterfall model is the earliest and is still commonly used. It focuses mainly on adequate support of management activities because it views the construction process as a sequence of actions from requirements analysis to the installation and maintenance of the product. Each of those actions ends on an intermediate document, a deliverable which is visible to management. As the field matured, the need for improving the software engineering process became apparent. The traditional view of the software process based on the waterfall model was criticised on several grounds [2]: it does not recognise iterations between stages; it tends to freeze the specification, increasing the maintenance costs; and it is difficult to incorporate new software engineering capabilities such as rapid prototyping, program transformation, and reuse. Furthermore, the waterfall model does not consider verification and validation as an integral part of the software construction process.

Alternative, radically different models were proposed with the common motivation of reducing the development costs, improving the reliability of software products, and most importantly, meeting

the ultimate users' needs. These models include rapid throwaway prototypes, incremental development, evolutionary prototypes, reusable software, and automated software synthesis (see [37] for a comparative review).

The model proposed by Boehm [2] is an extension of the rapid throwaway approach. The spiral model focuses on risk-mangement. Each step was expanded to include a validation and verification activity to cover high risk elements, reuse consideration, and prototyping. It differs from the document-driven models such as the waterfall model and the specification-driven model such as Lehman's two-leg model [3] by its *risk management plan.* The risk management plan includes identifying the top risk items of a given project, developing a plan for resolving the risk items, and assessing the project's progress through monthly reviews.

2.3 Problem Structuring

In order to reach an understanding of a complex business activity on which to base the analysis of the problem to be solved, some structure must be imposed. There are three underlying principles of structuring used during problem analysis [36]: partitioning, abstraction, and projection. Partitioning divides a problem into sub-problems (e.g. SADT). For example, the problem of monitoring patients in an intensive care unit may be decomposed into monitoring, and analysis sub-problems. These are not necessarily sub-systems. At the design stage, the components that make up the target system will have little or no relationships to these sub-problems. An abstraction decomposes a problem along an aggregation/specialisation scheme, so each part is an example of its parent and inherits all its features (e.g. Object-Oriented Analysis). For example, we may consider hospital staff and medical staff as instances of the abstract concept staff. A projection is defined as describing the system from multiple external viewpoints. For example, the monitoring sub-problem can be analysed from a doctor's viewpoint and from a ward nurse's viewpoint. CORE and SADT were the first methods to support some form of projection and partitioning.

2.4 'Viewpoints on Viewpoints'

A viewpoint can be informally defined as a perspective from which a domain can be observed. An analyst trying to find out how library resources are to be managed may get very different accounts depending on whether he talks to a borrower or to the librarian. By taking account of both views the analyst gets a better picture of the domain than by considering only one.

There are three types of viewpoints:

- The agent responsible for the viewpoint, i.e. the person observing the problem domain. The agent could be a user, an analyst, a domain expert, a designer, etc. A viewpoint method is called agent-oriented if it is based on this type of viewpoint.
- The process by which that part of the domain perceived by the agent is modelled. The process could be a set of correctness-preserving transformations, a set of elaborations (an elaboration does not have to be correctness-preserving), etc. A viewpoint method is called process-oriented if it is concerned with the properties of the viewpoint modelling process.
- The representation scheme in which an agent's perception is described. A description of the perception is called a view. A view could be a data/control flow diagram or a Z schema [6]. A viewpoint method is classified as scheme-oriented if it operates on the characteristics of the representation scheme used to describe a view.

This broad categorisation corresponds to the three areas of the single viewpoint requirements engineering: requirements acquisition, specification processes, and specification languages respectively. Multiple viewpoint approaches differ from the single viewpoint approaches in their explicit capture of alternative descriptions, whether it is a requirements specification, a system model, a domain model, or a cognitive model, and their support for resolving conflicts inherent in the process. The 'univocality' of single viewpoint approaches has been criticised by several authors (see for instance Easterbrook [45]).

The methods are analysed along the following lines:

- The definition of a viewpoint.
- The reasoning within individual viewpoints, e.g., to check the internal consistency of a view.
- The reasoning between different viewpoints, e.g., comparison of disparate views, conflict analysis.
- The revision of a viewpoint, e.g., the modification of a view to restore consistency, fitting in new information, or the creation of a new viewpoint.

2.4.1 Process-Oriented Methods

This class includes the works of Feather [49] and Robinson [102]. Feather and Robinson take the view that a software specification process begins with a trivially simple specification, incrementally elaborates it in a number of parallel "lines" of design, and merges the specifications that result from each of those divergent lines to achieve the fully detailed specification. A viewpoint is a line of design. An elaboration is a transformation that deliberately changes the meaning of the specification to which it is applied. A conventional transformation generally keeps the meaning of a specification constant, i.e. a correctness-preserving transformation.

The approach to merging different views is to "replay" the evolutionary transformations of the separate lines of design in a serial order. If the parallel evolutions are completely independent then the result of merging them will be the same regardless of the serial order followed. Consider, for example (from [49]), the case where there are two viewpoints and each viewpoint comprises a single evolution. In Figure 2.1 `s0` is the initial specification, `e1` and `e2` are the evolutions, giving rise to evolved specifications `s1` and `s2` respectively. To produce the specification that combines these elaborations, apply `e2` to the result of applying `e1` to `s0`, or apply `e1` to the result of applying `e2` to `s0`.

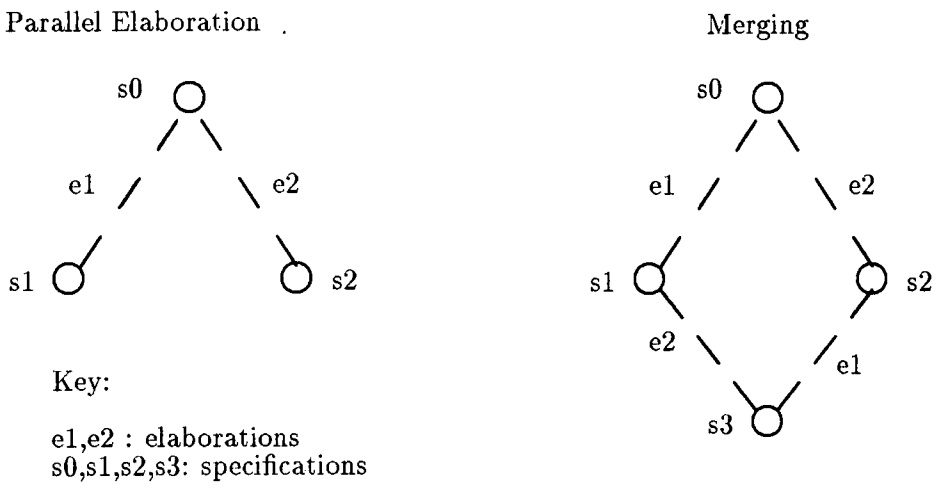

Figure 2.1: Merging two viewpoints

There are, however, cases where evolutions interfere with one another in some way. For example, one evolution renames a function `F` to `G`, while another evolution extends function `F` with an extra formal parameter. This interference can be resolved by applying the extension before the renaming. The merging process consists of interference detection and interference resolution.

The process of detecting interferences is based on the properties of the specification and the changes that affect those properties. The specification properties considered are limited to *terminology* (the set of signatures of the specification's constructs, e.g. a data structure) and *usage* (the use that constructs make of one another, e.g. a reference to a data structure). The detection of interferences consists of two stages:

- Determine the effects that each of the evolutionary transformations induces on each specification property. A possible change to terminology is 'rename the parameter named p of construct c to p" and a possible change to usage might be 'add to construct c a use of construct d'.
- Make pairwise comparisons of changes within each property, and of changes between each property.

The following are examples of classes of interferences, between the terminology changes, considered

together with their possible resolutions:

- Duplication: The two transformations make the same terminology change. Only one of the two transformations needs to be applied.
- Renaming interference: One transformation renames something that the other transformation refers to. A possible resolution is to apply the renaming transformation second.
- Duplication with renaming: For example, the transformations rename the same construct to different names. Only one transformation is applied.
- New name clash: The transformations introduce the same name for different purposes. For example, adding different constructs with the same name. This is resolved by replacing the new name introduced in one of the transformation with a different name.
- Remove and modify: One transformation removes a construct that the other modifies (e.g., adds a new parameter to). The interference is resolved by applying the modification transformation before the removal transformation.
- Contradiction: The transformations cannot both be performed. For example, the transformations change the type of the same parameter to different types. There is no resolution method for contradictions yet.

The interferences between the usage changes are classified similarly to the interferences between the terminology changes. For example:

- Duplication: The transformations duplicate the same usage change. For example, one transformation adds to construct `c1` the use of construct `d1` and another transformation adds to construct `c2` the use of construct `d2` while `c1 = c2 & d1 = d2`. Only one of the transformations needs to be applied.
- Add & Modify ordering dependency. For example, one transformation adds to `c1` the use of `d1` and the other removes all `c2`'s uses of `d2` while `c1 = c2 & d1 = d2`. This conflict can be resolved by applying the modification to the added use or by applying the addition only.

The incremental approach allows Feather to maintain a specification by altering elaborations (the process that we call the revision of a viewpoint) and then "replaying" them to create a new specification. Each elaboration is recorded in terms of the changes it induces on the specification properties.

Robinson also uses the parallel elaboration approach but the starting point for the elaborations is a goal tree which stores different levels of domain goals. The attributes of a domain goal are instantiated to different perspectives. For example, in an academic library domain the `proposed value` attribute of the goal `loan period` can be set to '2 weeks' from a library staff's perspective and to '6 months' from a library user's perspective. Once a perspective is created, the specification construction process can take place in the same way as Feather's: the perspectives are operationalised by applying a sequence of elaborations to create specification components for each line of design. The goal perspectives, the resulting specifications, and the elaborations are recorded together with the links between the goal perspectives and the specification components that support them.

The integration of the resulting specifications involves the following steps:

1. Correspondence Identification to isolate equivalent specifications. Specifications can only be compared if there is some similarity between them. The analyst is relied on to carry out this process.

2. Conflict Detection and Characterisation. Syntactic differences between specification components are mapped to differences of domain goal attribute evaluations from which the components were derived. This is also carried out by an intelligent agent.

3. Conflict Resolution. When a conflict is detected, the elaboration links are traced up to identify the domain goals from which the conflicting specifications were derived. So the resolution process concentrates on removing conflicts between domain goal perspectives. The conflict resolution method is based on the attributes utility theory. Attribute utility analysis provides a way for comparing alternatives with varying attribute values in order to pick the one that offers the maximum overall utility. Robinson uses the domain goal attributes for

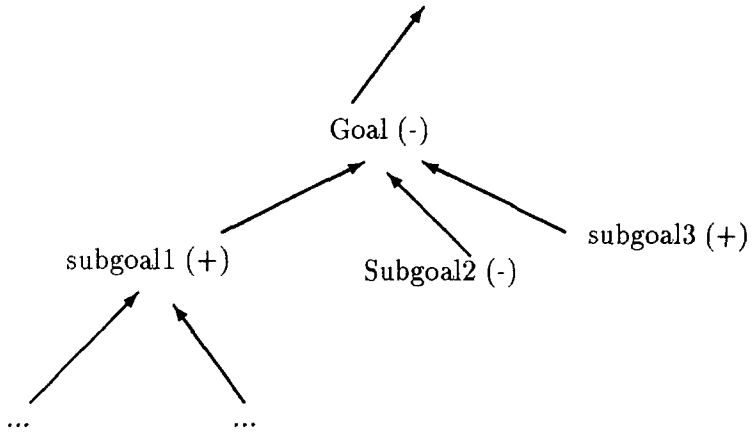

Figure 2.2: Domain goal relationships

developing compromises. Figure 2.2 shows how the goal/subgoal structure is represented in Robinson's model. A "+" sign associated with a subgoal indicates that an increase in satisfaction of the subgoal will contribute an increase in satisfaction of its parent goal. A "-" sign indicates a reverse correlation between the satisfaction of the subgoal and its contribution to the satisfaction of its parent goal. The role of these indicators is to guide the search for the maximum satisfaction of a goal during conflict resolution. The utility (degree of satisfaction) of a goal may be a function of the goal's importance in the domain, relative subgoal contributions, proposed values, feasibility, etc. The attribute values can be modified to find a compromise in case of an impasse. The utility function is undefined in Robinson's model and the control mechanism for generating resolutions is deferred as a difficult problem.

4. Resolution Implementation. Changes made at the goal level, as a result of conflict resolution, should be mapped back to the specification level. Similarly to Feather's model, the elaborations are replayed with the new domain goal attributes to produce a new specification.

There is no evidence that the quality of the specification produced using Feather's model will be better than a specification produced by a single line of design. That is, the model does not allow for the validation of the specifications. Robinson's model improves on Feather's by incorporating domain modelling, so a specification component can be 'justified' in terms of the domain goals from which it originates.

2.4.2 Scheme-Oriented Methods

A typical example of a scheme-oriented method is the work of Niskier *et al.* [90]. A viewpoint refers to a particular formalism that focuses on a particular aspect of a system description. For example, Data Flow Diagrams, Entity Relationship models and Petri Nets are better suited to describe functional, informational, and operational aspects of a system respectively. A viewpoint captures syntactic and semantic properties of a representation scheme. The syntactic properties are related to the correct combination of the scheme's primitive elements. For instance, every process in a data flow scheme should have at least an input and every event in a petri net scheme should have at least a precondition. Semantic properties capture the expected behaviours of the specified system. For example, "if a dataflow is an output of a process and an input to a file, then the process is updating the file".

A knowledge-based system called PRISMA (Portuguese for prism) is constructed to support the construction and integration of different views. Reasoning in PRISMA is based on a set of heuristics that use the properties of the schemes involved. Given a view, the following checks can be performed in the PRISMA environment:

- **Agenda generation.** The agenda mechanism is driven by the structuring heuristics. The structuring heuristics operate on the syntax properties of the representation used to characterise unsatisfactory situations in a view and to provide advice on how to overcome these problems. An example of a structuring heuristic for a Petri Net view is:

  ```
  If there is an event with only preconditions, then define its
  postconditions looking for the conditions identifying the state
  of the system after the event occurs.
  ```

- **View validation.** The view validation is based on the validation heuristics that operate on the semantic properties of the representation scheme. The role of the view validation is to check the 'correctness' of view by paraphrasing, i.e. generate natural language descriptions of some aspects of the specification.

- **Complementary checking.** The goal of Complementary checking is to ensure partial consistency of different views. Complementary checking is driven by the complementary heuristics which are pre-defined mappings, relating properties of one view to the corresponding properties of another view. For example, to each process representing a data transformation (Data-Flow Diagram) there is an associated event representing the occurrence of that transformation (Petri-net).

By using complementary heuristics the PRISMA approach suppresses conflicts, and does not, therefore, profit from using multiple viewpoints. In addition the authors avoided the 'correspondence' problem by selecting representation schemes that have correspondence. For example, it is difficult to integrate object-oriented models with data flow diagrams using PRISMA.

Finkelstein [57] considers the 'multiple perspective problem' in the wider context of "programming-in-the-large", an activity which involves many participants with different skills, roles, knowledge and expertise. Each participant has differing perspectives on, and about knowledge of, various aspects of software development and the application area. Further, the knowledge within each perspective may be represented in different ways and the development may be carried out concurrently by those involved using different development strategies at different stages of the development. Finkelstein uses viewpoints to partition the system specification, the development method and the formal representations used to express the system specification. A viewpoint is defined as *a combination of the idea of an "actor", "knowledge source", "role" or "agent" in the development process and the idea of a "view" or "perspective" which an actor maintains. In software terms it is a loosely coupled, locally managed object which encapsulates partial knowledge about the system and domain, specified in a particular, suitable representation scheme, and partial knowledge of the process of design.* Each viewpoint is composed of the following slots:

- a representation **style**, the scheme and notation by which the viewpoint expresses what it can see;
- **a domain**, which defines that part of the "world" delineated in the style;

- **a specification**, the statements expressed in the viewpoint's style describing particular domains;
- **a work plan**, describing the process by which the specification can be built;
- **a work record**, an account of the history and current state of the development.

The work plan is the most important and complex slot in a viewpoint. A work plan is composed of four 'sub-slots':

- the **assembly actions** slot which contains the actions available to the developer to build a specification;
- the **check actions** slot which contains the action available to the developer to check the consistency of the specification;
- the **viewpoint actions** slot which creates new viewpoints as development proceeds;
- the **guide action** slot which provides the developer with guidance on what to do and when.

There are two types of check actions:

- **in-viewpoint checks**, check the consistency of the specification within the viewpoint;
- **inter-viewpoint checks**, check the consistency of the specification with those maintained by other viewpoints. Inter-viewpoint checks are, in turn, of two types: "transfer" and "resolve" corresponding to cooperation and competition respectively.

2.4.3 Agent-Oriented Methods

This category includes the *viewpoint analysis* method proposed by Leite [72], The CORE method [88], the Viewpoint Oriented Approach [68], the dialogue model introduced by Finkelstein *et al.* [56], and Easterbrook's 'multiple perspective' model [45].

Finkelstein defines a viewpoint as a participant in the dialogue responsible for maintaining a particular perspective. A perspective can correspond to the participant's role in the application domain or to an area of concern to that participant. As an agent can hold several responsibilities he can hold several viewpoints. Requirements engineering through dialogue takes the form of a game, in which moves consist of speech acts, such as assertion, question, challenge, or withdrawal, and a set of rules to maintain a 'legal' dialogue. Viewpoints are committed to anything they state and to anything stated by other viewpoints. They are responsible for the maintenance of the validity and the internal consistency of their views, and through the rules defined in the dialogue model, of the other views by consulting the chains of reasoning made by other viewpoints, and by requesting information which they need to verify the conclusions.

Easterbrook's work is based on the dialogue model. Easterbrook interprets the dialogue (between an analyst and an information source) transcripts into some formal language (first order predicate calculus). The first task is to break the textual information into chunks, where each chunk focuses on a particular area of knowledge (or a topic) called a perspective. Each chunk is identified by its source, where a source could be a person or a group of people, and then interpreted into a set of propositions which act as a formal representation of the information contained in that chunk. The formal representation of a perspective is called a viewpoint. If a viewpoint becomes inconsistent it is split into consistent sub-viewpoints creating new topics, i.e., conflicting statements are placed in separate descendants of the current viewpoint. The explorative nature of viewpoint decomposition is similar to the issue-based approach (e.g., [70]). In this way, the viewpoints descriptions are built up through the addition (assertion) or removal (retraction) of statements (called commitments). The *commitment reasoning scheme* allows a statement to be in one of four states: Uncommitted state is the default, indicating that the item has not been discussed yet. The true and false states indicate that a person has committed himself to one or the other. The inconsistent state is used when a person has contradicted himself. Conflicts detection is based on the detection of logical inconsistencies in the first order predicate calculus scheme, although the model allows other representations to be used given that inference rules are provided.

Part of Easterbrook's model is the *computer-supported negotiation* model supported by a tool that

provides clerical support and some guidance for the participants, allowing them to compare their descriptions and negotiate options for resolution. Given two descriptions, within which particular statements are known to conflict, the participants should:

- establish correspondences between the two descriptions by comparing the statements around the conflicting ones in order to establish a context for the conflict. The result is a list of correspondences between items in the viewpoints and a list of specific disparities between items.
- identify the conflict issues (the points to be addressed). An example of an issue is the loan period of books and the fines policy issue.
- agree a resolution criteria by which to judge the possible resolutions with reference to the participants' satisfaction.
- generate resolution options. The model is restricted to three types of conflicts: conflicts in terminology, e.g. the same terms used for different concepts, conflicting interpretations, and conflicting designs assuming that requirements contain some design information. The conflicts are given values reflecting their degree of severity: non-interference, partially-interfering, and mutually exclusive.
- select a resolution from the options available.

CORE [87, 88, 69] was developed for requirements engineering with interactive elicitation from multiple requirements sources as its primary aim [88]. Viewpoints are seen as agents that have interests to be supported/influenced by the proposed system and act as points where information elicitation takes place ("possessors of requirements" [69]). By virtue of its support for both projection and decomposition CORE identifies two types of viewpoints: bounding viewpoints and defining viewpoints. Bounding viewpoints are the external agents that interact with the target system (called environmental agents) whereas defining viewpoints are the functional processes that make up the target system. For a patient-monitoring system, for instance, a hospital staff member such as a ward nurse, medical staff member such as a doctor, the central station, the bed and patient may

be identified as bounding viewpoints and defining viewpoints are analysis and monitoring. CORE assists in meeting the following objectives:

- obtaining information from viewpoints (who have only as yet only half-formed ideas about the service required from the proposed system);
- detecting and illustrating differences in perception of the required service;
- getting decisions about whose view is to prevail or aiding the development of compromises;
- achieving completeness and consistency of the specified information, where possible, and a record of each instance where it is not achieved;
- recording it in a form understandable to the viewpoints and usable for developing a formal specification of the system requirements, suitable as a contract to develop the proposed system.

CORE comprises the following steps:

1. Viewpoint identification and structuring - classification of viewpoints.
2. Information gathering - interviewing each viewpoint to identify the actions performed by that viewpoint, the actions the proposed system is required to perform for the viewpoint, and their production and consumption of data flow from other viewpoints.
3. Data structuring - construct a diagram resembling a Jackson structure diagram [63] which shows the legal sequencing of the output data flows recorded during information gathering.
4. Actions structuring (isolated) - using the actions and their interfaces from information gathering and the order of derivation of the outputs from Data Structuring, actions structuring (isolated) consists of establishing dependencies among the actions and producing a Single-Viewpoint Model similar to a data flow diagram, for each viewpoint.
5. Actions structuring (combined) - constructs Combined-Viewpoint Models. A combined-viewpoint model (or a transaction) is typically a small set of interconnected actions, from

different viewpoints, which interact closely to perform some specific sub-tasks of the system in its environment.

6. Constraints analysis - once the individual viewpoints have been completely reconciled transactions are 'animated' through 'what-if' enquiries to discover anything that may cause a problem leading to a break-down or failure to provide the required service within the defined constraints. For example, the analyst might ask: If iteration is involved, could there be convergence problems or error build-up?

These activities are driven by a set of heuristics that are hints about checks that should be performed at each step. These heuristics can be seen as a special case of the heuristics built into PRISMA when using a data flow-based representation only and employing animation [69] instead of paraphrasing for specification validation.

In **viewpoint analysis**, proposed by Leite [72], multiple analysts (viewpoints) describe their understanding of a problem in the same universe of discourse in the same language VWPL (VieWPoint Language) using a common vocabulary. Each analyst constructs his view using three perspectives corresponding to the modelling aspects: data modelling, process modelling, and actor modelling. Actor modelling is related to the agents responsible for the processes. To attach some semantics to the information encoded in the viewpoint language, viewpoints use two hierarchies: the "is-a" hierarchy to represent specialisation relationships between keywords; and the "parts-of" hierarchy to represent decomposition relationships. Leite contends that the heavy use of redundancy will improve the chances of detecting problems related to consistency and completeness. Each viewpoint then integrates the perspectives into a view, resolving the internal conflicts. Once the views are completed they are compared, producing a list of 'discrepancies' that acts as part of an 'agenda' for negotiating resolutions to conflicts between viewpoints. Viewpoint analysis is critically evaluated in the next chapter.

Kotonya and Sommerville [68] proposed a Viewpoint-based Object-oriented Approach to requirements analysis (VOA) in which a viewpoint is seen as *an external entity that interacts with the*

system being analysed, but one that can exist without the presence of the system. VOA is two-layered: the viewpoint layer, concerned with the behaviour of the environment of the proposed system, and the system layer, concerned with the system's responses to its environment *. VOA includes four main stages:

- viewpoint identification;
- viewpoint structuring and decomposition;
- information collection;
- reconciliation of information across viewpoints;

The best way to explain VOA is to compare it to CORE given the close similarities between the two. VOA and CORE share:

- classification of the external entities: direct and indirect viewpoints (CORE), and active and passive viewpoints (VOA).
- establishment of a viewpoint structure but use different structuring schemes. CORE employs functional decomposition (role/sub-role) whereas VOA provides for inheritance (abstraction/ specialisation)
- explicit capture of the interactions between entities in the environment and the target system in terms of the services the system is required to provide and the constraints under which the services should be provided.

The two approaches differ in the following:

- CORE's viewpoint structure is a mixture of external and internal viewpoints with the top level defined as 'system + environment' † while VOA treats them separately;

*VOA is restricted to the viewpoint layer

†this feature of CORE has been described in [68] as confusing due to its poorly defined notion of a viewpoint

- CORE captures the interactions between the external viewpoints for a fuller model of the environment;
- VOA distributes the non-functional constraints across the viewpoint structure and reconciles them across the viewpoints while CORE treats them in a separate activity (constraints analysis) after the full viewpoints have been integrated;
- CORE provides heuristics for detecting structural inconsistencies. VOA does not provide a firm mechanism for 'information reconciliation across viewpoints'.

2.5 Summary

Multi-viewpoint approaches differ from the single-viewpoint approaches in their explicit capture of alternative descriptions, whether it is a requirements specification, a system model, a domain model, or a cognitive model, and their support for resolving conflicts inherent in the process. The 'univocality' of single viewpoint approaches has been criticised by several authors (see for instance Easterbrook [45]) for their suppression/avoidance of conflicts. As Elam *et al* [46] point out, one goal of a software design task should be not the minimisation of conflicts, but rather the identification (or surfacing) and subsequent resolution of conflicts. In their study of conflict behaviour in the requirements engineering phase of large systems development, Elam *et al* have shown that conflicts or interpersonal disagreements increase the quality of group decision making by stimulating critical thinking, increasing group involvement, and widening the search for alternatives. Conflicts resolution has the following features [43, 111]:

1. The iterative nature of the process. Negotiators often employ an iterative strategy of generation followed by evaluation.
2. The participative nature of the process. All the viewpoints should be involved in the reconciliation process.

3. The learning process involved. A participative framework intends to encourage a pooling of knowledge and insight, and the decision-makers become engaged in a process of learning and understanding.

4. The amount of information to be handled in order to make a reasonable decision.

A viewpoint method is identified as agent-oriented, scheme-oriented, or process-oriented depending on the type of viewpoint it adopts (agent, process, or formalism). The following issues are addressed by viewpoint methods:

- Viewpoint identification - there is an infinite number of angles from which a domain can be observed. A viewpoint method should have clear criteria for distinguishing viewpoints, i.e., an unambiguous definition of a viewpoint. Some methods fix a set of pre-defined viewpoints (e.g., Leite, CORE) while others begin with a set of viewpoints then identify others during the analysis process through decomposition, refinements, etc. (e.g., Easterbrook).

- View modelling - once a viewpoint has been identified it can be applied to the domain under analysis to produce a description of the domain from that viewpoint, i.e., a view. A method can either model views independently (competitive) or derive one from another (cooperative).

- Comparing disparate views - some methods compare the views in parallel with the view modelling process, others compare them only when they are 'final' (most of the methods). The result is a list of 'discrepancies'. Comparison makes sense only when the viewpoints correspond (have something in common). Establishing correspondence is a tough problem and none of the authors has a solution. Leite, for example, avoids the problem by making unrealistic assumptions (viewpoints have to observe a domain, almost from the same angle): viewpoints should consider the same topic, use a common vocabulary, and use the same language, VWPL which constrains how the rules should be expressed. Easterbrook assumes that viewpoints will not be wholly unfamiliar with other viewpoints' knowledge, so that they will be able to suggest correspondences between their views.

- Conflicts characterisation - the discrepancies resulting from the comparison are, often, syntactical differences, structural differences, or differences of terminology (viewpoints use different terminology to describe the same thing). Conflicts characterisation establishes an agenda to be used as input to the negotiation. Part of the negotiation process is to distinguish between real and apparent conflicts by exploring the context of the differences and gathering more information necessary to identify misunderstandings, differences in terminology, etc. [45].

- Conflicts resolution - once the different options about an issue have been identified a conflict resolution process is launched. Only Easterbrook and Robinson have attempted to address the conflict resolution problem. They both employ an iterative strategy of generation followed by evaluation. Sycara [111] uses this approach in her model of an automated labor mediator. Options for a resolution are suggested then evaluated against the satisfaction of the participants. The process is repeated until a reconciliation is achieved.

With the exception of Robinson and Easterbrook no approach has attempted to address the issue of conflict resolution, a crucial part of a multi-viewpoint method.

Chapter 3

Viewpoint Analysis: a Critical Evaluation

Leite proposed *Viewpoint Analysis* as a method for early requirements validation. He contends that by describing the same domain, from two different perspectives, in the same language using the same vocabulary and then examining the differences between the resulting descriptions the chances of detecting problems are 'enhanced'. Leite claims that his method provides a better approximation between the universe of discourse and the gathered facts (see chapter 1) than the existing methods. This chapter evaluates Leite's method from the early validation perspective, as well as from the perspective of multiple viewpoints usage.

3.1 Introduction

Brackett [28] describes Leite's thesis as 'valuable to anyone working seriously in developing requirements definition methods'. However, Leite's actual contribution needs careful examination from the

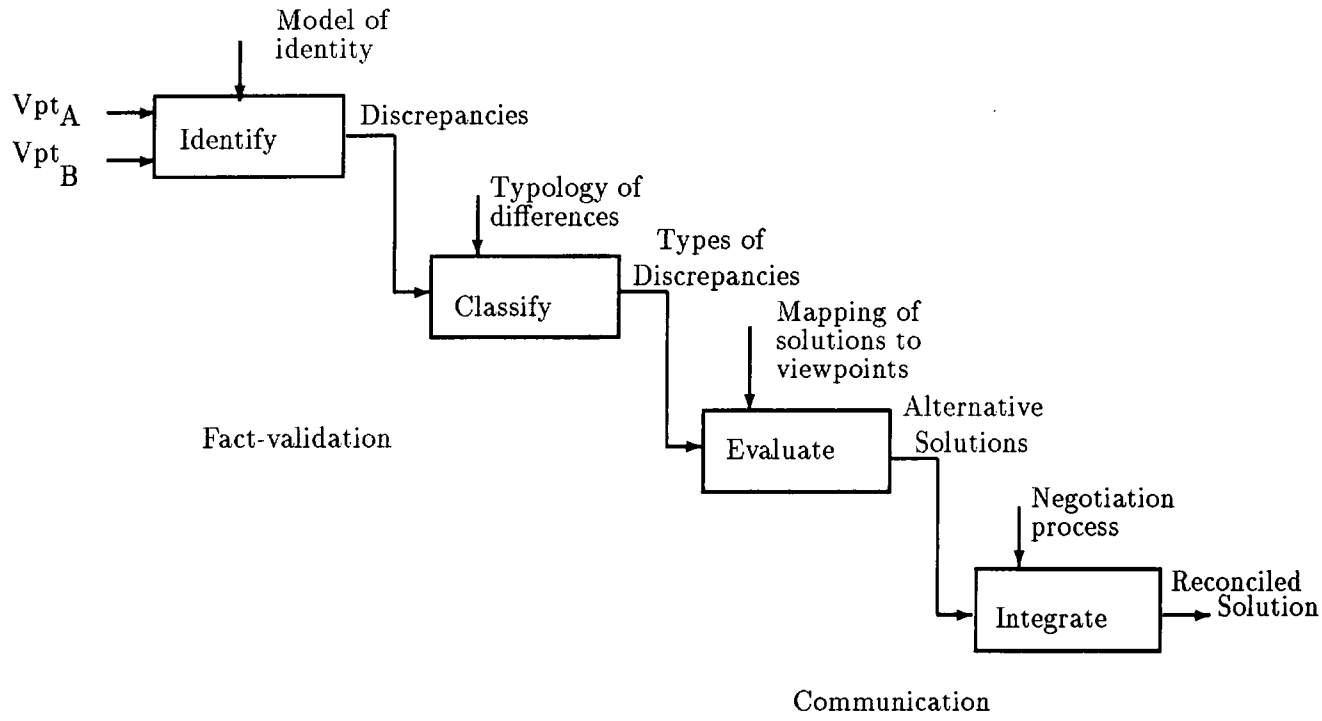

Figure 3.1: Viewpoint Resolution

point of view of requirements validation. Figure 3.1 represents a SADT (Structured Analysis and Design Technique [104] model of Leite's viewpoint resolution method. Leite's main work, however, is concentrated on the first part of the method, viewpoint analysis to support fact-validation. The second part, viewpoint reconciliation, to support communication, is deferred as a difficult problem. Viewpoint analysis consists of:

1. view construction, with the following guidelines:
   - find the facts
   - express the facts using the keywords of the application domain
   - classify the facts into object facts, actions facts, and agent facts
   - codify the facts using the VWPL language
2. static analysis

- compare (syntactically) the different descriptions (from the same viewpoint) then compare the different descriptions from different viewpoints
- classify the differences into missing information and wrong information as well as inconsistencies.

3.1.1 View Construction

In order to construct a view, an analyst (viewpoint) finds the facts using his favourite methods, representations and tools then describes the problem using three perspectives and two hierarchies. The perspectives are the actor perspective, the data perspective, and the process perspective. An example of perspective modeling is CORE's 'Data Structuring' and 'Action Structuring' used to model viewpoints, and SADT's datagrams and actigrams. The idea behind actor modeling is to model using the perspectives of those who are responsible for the processes, i.e., human agent or devices. The hierarchies are the is-a hierarchy and the parts-of hierarchy. A view is the integration of perspectives and hierarchies. The construction of a perspective is based on the assumption that the analysts use the application vocabulary (keywords of the application domain) when describing their views.

A perspective is expressed in the VieWPoint Language (VWPL). VWPL is a rule-based language with a predefined structure for the construction of rules. A rule is made of facts (a fact is a relationship between keywords of the application domain). A fact is composed of a fact-keyword and a fact-attribute. For example, the fact `(book =book-id =author =title)` has `book` as the fact-keyword and has as attributes `book-id`, `author`, and `title`.

Facts are classified according to *type* and *class*. There are three types of facts: the object fact, the action fact, and the agent fact. Classes are the different roles each fact may play in a rule. In a rule, a fact can:

- be deleted from the working memory (called the input class),

- be added to the working memory (the output class), or
- remain in the working memory (the invariant class).

For each type of perspective there is a special combination of types and classes. For the process perspective, for instance, the rule structure is as follows:

- LHS - **input** is an action and objects (optional), **Invariant** can be an agent and/or an object.
- RHS - **output** is an object.

For example, the action "check-out a copy of a book" from a library problem can be represented by the following rule [73]:

```
(21 ((check-out =borrower =copy)
  (book =author =title =copy)
  (library-copy =copy)
  (<not> (copy-borrower =borrower =copy)))
---->
  (($delete-from wm (check-out =borrower =copy))
  ($add-to wm (copy-borrower =borrower =copy))))

(is-a (1 (user borrower library-staff)))
(parts-of (2 (book author title copy)))
```

The fact *delete-from* the working memory (RHS) is the pre-condition **input** (LHS). A pre-condition **input** is discarded in order to maintain the working memory consistency. The fact *add-to* the working memory (RHS) is the post-condition **output**. The other facts in the LHS are the pre-conditions that did not change, that is, the **invariants**

In rule 21 **check-out** is the input, **book, library-copy** and < *not* > **(copy-borrower)** are

invariants and **copy-borrower** is the output. The attributes are **borrower, copy, author, title,** and **copy.**

In the hierarchies *is-a* and *parts-of* the head of the list is the root of the hierarchy. In the **is-a** hierarchy of rule 21 the agent **user** is the higher generalisation of **borrower** and **library-staff.** In the **parts-of** hierarchy: **author, title**, and **copy** are parts of the object **book.**

For an actor perspective the rule structure is as follows:

- LHS - **input** is an agent, **invariant** can be an object and/or and agent (both optional).
- RHS - **output** is an action.

3.1.2 Static Analysis

Static analysis is the syntactic comparison of different perspectives and different views using pattern, and partial matching. Static analysis has two tasks: finding which rules are similar, and, once rules are paired, identifying and classifying the discrepancies between them (see figure 3.2). The discrepancies are classified as follows:

- Incorrectness: contradiction between facts of the different rule sets.
- Incompleteness:
  - missing rules:
  - missing facts
  - incomplete hierarchies with respect to rule facts
- Inconsistency
  - contradiction between a fact and the hierarchy
  - redundancy in the same rule set.

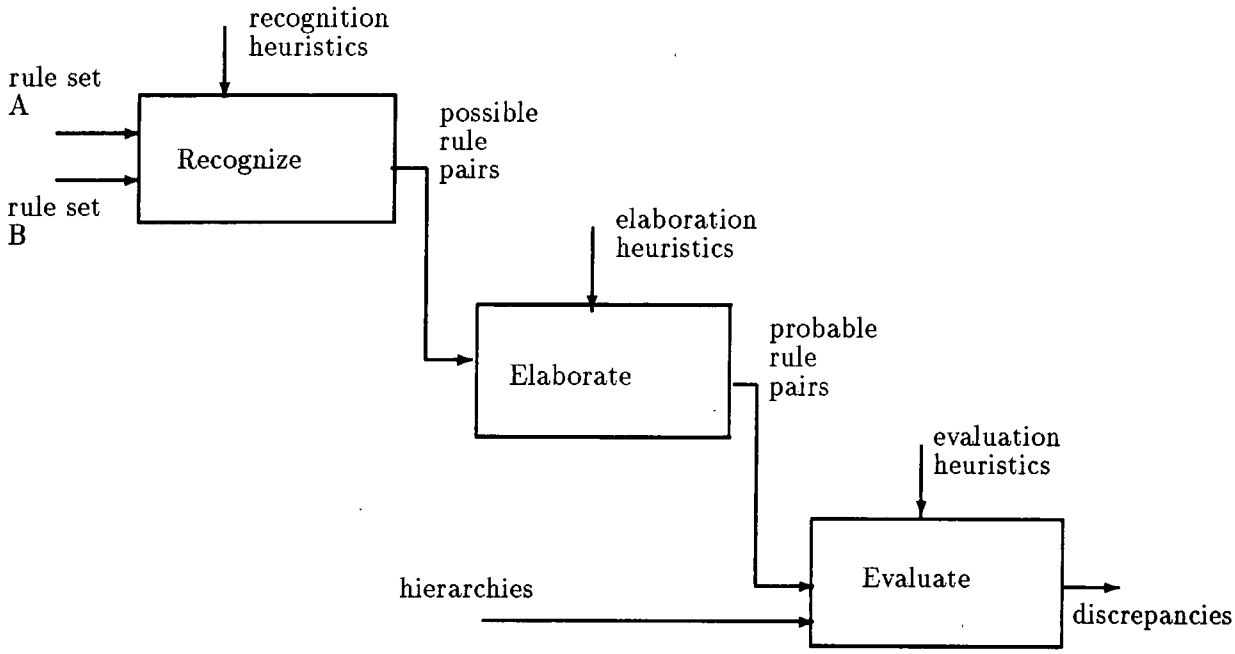

Figure 3.2: The static analyzer heuristics

Static analysis is driven by a set of heuristics built into a support tool called *static analyser*. There are four types of heuristics used in the static analyser (see Figure 3.2): The partial matching heuristics are used in finding out similarities between facts from the different rule sets. The scoring heuristics represent a scoring scheme for compounding different matching scores to find out the the possible rule pairs. The scoring evaluation heuristics are used to identify the best pairings between rules. The classification heuristics use the hierarchies to define the types of the critics produced: missing information, wrong information, or inconsistencies. For example [74][page 1263]:

```
If a fact in rule x from viewpointA
     is a leaf in a parts-of hierarchy
And a fact in rule y from viewpointB
      is a leaf in a parts-of hierarchy
And the hierarchy root is the same.
        (leaves of the same hierarchy)
```

```
--->
the facts are in contradiction.
        (one of the rules has wrong information)
```

3.2 The Limitations of the Leite Method

The shortcomings of the viewpoint analysis method can be summarised in the following points:

1. The previous chapter concludes with the observation that under the severe restrictions imposed by Leite in his method (viewpoints should consider the same topic, use a common vocabulary, and use the same language, VWPL which imposes constrains how the rules should be expressed) the viewpoint analysis is almost reduced to a single perspective method in which different viewpoints must observe a domain almost from the same angle. But in evaluating his method, Leite acknowledges that the assumptions need to be relaxed. He suggests that a negotiation process between the viewpoint holders should take place before the application of his method; i.e., that the facts should be analysed before their codification into VWPL. In other words, a sort of viewpoint resolution process should take place before the the application of his method.

2. The method is is syntax-oriented. Using the method, similar concepts with different syntactic forms can be declared different or even contradictory, and different concepts with similar syntactic forms can be found to be the same.

3. The method cannot say anything about two statements, considered to be consistent, but which are not correct with respect to the universe of discourse.

4. The method does not have a resolution/reconciliation strategy and it is difficult to 'retrofit' a resolution strategy in the technique's present form(one may draw an analogy with specification reuse [55]). The main reasons for the difficulties are:

(a) The absence of a strategy for identifying the viewpoints' holders (actors), necessary to identify the universe of discourse within which the facts are assessed.

(b) The absence of explicit links between the universe of discourse and the gathered facts (except the assumption that the application vocabulary is used in the process of codifying the facts), necessary for validation (links can be used to trace the roots of conflicts back in the universe of discourse [45, 102, 51]).

(c) The non-involvement of the users directly in the validation process. The method is oriented to those who build the requirements model, who are usually the people on the supply side. This leads to uneffective communication. Communication plays an important role in conflicts resolution [46].

3.3 Application of the Method

Since there are no restrictions on what representation should be used to record the facts before their codification into VWPL, we assume natural language is the 'natural' choice. Two counter examples to Leite's strategy are chosen: the first represents two views with similar meaning but with slightly different syntax. The second example represents two views with different meanings but with similar syntax. The texts were obtained from two different portions of specifications produced by CORE and Remora respectively [54]. CORE enables the reader to trace the derivation of each dataflow-fragment from the text representing the needs statement because each such fragment is indexed by line into the text. The CORE specification is then used to guide the derivation of the second text from the Remora specification.

The first example concerns a system for on-line monitoring of patients in an intensive care unit. The following are two possible views about the initial activation of the monitoring system.

View_One

When a new patient is admitted a hospital staff member activates the monitoring

system. The monitoring system then prompts the staff to set up the initial patient's data.

View_Two

when a new patient is admitted a hospital staff member is responsible for activating the monitoring system and initialising the patient's data.

The following are possible representations of the views in VWPL:

```
            Viewpoint_One
1 (activate-system =patient-admitted) (nurse =status)
(system = patient-monitoring-system)
------->
$delete-from-wm (activate-system =patient-admitted)
$add-to-wm (activate-request =initial-prompt)

  : activate-system (of type action) is the pre-condition input
  : nurse (agent) and system (object) are invariants
  : activate-request (object) is the post-condiction output
  : patient-admitted, status, patient-monitoring-system,
  : and initial-prompt are attributes.

2 (set-up-patient-data =patient-identity =initial-prompt)
(nurse =status)
(patient-admitted =patient-identity)
(system = patient-monitoring-system)
$delete-from-wm(set-up-patient-data =patient-identity=initial-prompt)
$add-to-wm (patient-data =initial-patient-data)

      Hierarchies:
```

```
    parts-of (patient-data patient-identity initial-patient-data)
    is-a (staff nurse)

  : patient-identity initial-patient-data are parts of
  : patient-data
  : staff is a generalisation of nurse

            Viewpoint_Two
10 (activate-monitoring-system =new-patient-admission)
(medical-staff-member =name =rank)
(monitoring-system)
------>
$delete-from-wm (activate-monitoring-system = new-patient-admission)

20 (initialise-data =new-patient-admission)
 (medical-staff-member =name =rank)
------>
$delete-from-wm (initialise-data =new-patient-admission)
$add-to-wm (patient-data =patient-id =initial-unsensed-parameters
 =Upper-bound-monitored-params = etc)

       Hierarchies:
    is-a (staff medical-staff-member hospital-staff-member
    parts-of (patient-data patient-id initial-unsensed-parameters Upper-bound-monitored-p
```

Using Leite's static analysis the possible rule pairs are (Rule1,Rule10) and (Rule2,Rule20). Comparing Rule1 and Rule10 syntactically we obtain the following discrepancies:

1. the facts *(nurse =status)* from Rule1 and *(medical-staff-member =name =rank)* from Rule10

are in contradiction because they are of the same type and of the same class (agent-invariant), syntactically different and have the same root in the *is-a hierarchy*

2. the attributes of *system* in Rule1 and *monitoring-system* in Rule10 do not correspond while they syntactically match.

3. the facts *(activate-system =patient-admitted)* and *(activate-monitoring-system = new-patient-admission)* are a perfect match.

4. Rule10 is missing the information *(activate-request =initial-prompt)*

5. the facts *(nurse =status)* from Rule2 and *(medical-staff-member =name =rank)* from Rule20 are in contradiction

6. Rule20 is missing the fact *(patient-admitted =patient-identity)*

7. the attributes of *patient-data* in rules Rule2 and Rulr20 do not correspond

The second example concerns a system for monitoring a high security intensive care unit. The monitoring system should poll sensors at regular intervals. When sensors detect undesirable events the system notifies the emergency services of exceptional conditions. The following are two possible interpretations of the problem:

Text_Three

A system for monitoring patients polls sensors every 60 seconds. If the sensors detect abnormalities the system notifies the emergency services of exceptional health conditions.

Text_Four

A system for monitoring alarms polls sensors every 60 seconds. When the sensors detect abnormalities the system notifies the emergency services of exceptional security conditions.

```
            Viewpoint_One
3 (notify =alarm-buzzer-sounding) (exceptional-condition =patient)
(patient-monitoring-system =monitoring-data)
(sensor =location =type =60-seconds)
------>
$delete from wm (notify =alarm-buzzer-sounding)
$add to wm (alarm =alert-alarm-state =audi-visual-alarm)
           (emergency-services =health}

Hierarchies
(is-a (emergency-services doctor)
(parts-of (monitoring-data blood-pressure temperature))


            Viewpoint_Two
4 (notify =alarm-sounding) (exceptional-condition =alarm)
(alarm-monitoring-system =monitoring-data)
(sensor =location =type =60-seconds)
------>
$add to wm (alarm =audi-visual-alarm) (emergency-services =security)
$delete from wm (notify =alarm-sounding)

Hierarchies

(is-a (1 (alarm-state alert-alarm-state safe-alarm-state)
         (emergency services police firemen)
(parts-of (monitoring-data lights)
```

Similarly, applying static analysis heuristics to the rules 3 and 4 the following observations can be

made:

- the rules 3 and 4 should, according to the method, be declared as 'missing rules' because they are are referring to two different parts (patient monitoring and alarm monitoring) of the same system, but
- the heuristics hinted at some 'discrepancies' between the two views. For example, the attributes of the facts `(exceptional-condition =alarm)` and `(exceptional-condition =patient)` do not correspond.
- the fact 'the monitoring system polls sensors every 60 seconds' is consistent with the rest of the facts but is 'critically' wrong.

3.4 Summary

Leite uses the viewpoint approach to support an early requirements validation by pointing out discrepancies in a problem description. The viewpoint analysis method is evaluated from the early validation perspective, as well as from the perspective of multiple viewpoints usage. As we have shown in this chapter each limitation of the viewpoint analysis method has been reflected in the quality of requirements validation; i.e., the type and quality of the discrepancies, in understanding a problem, the method can point out and help resolving them. It has been shown, through two case studies, the limitations of a syntactic analysis of information; the absence of a strategy for identifying viewpoints (actors) that make up the universe of discourse lead to the absence of a universe of discourse within which information is validated. This in turn lead to the unability of the method to detect incorrect information with respect to the universe of discourse and to the absence of explicit links between the gathered information and the universe of discourse necessary for traceability. The absence of a conflict resolution strategy and the non-involvement of the users directly in the validation process does not help communication, necessary for resolving open issues.

The hypothesis of this thesis is that by improving the viewpoint resolution process, as proposed by Leite, the validation process could be improved as a result.

Chapter 4

Natural Language for Requirements Engineering

4.1 Natural Language Engineering

Understanding a text by a computer means translating it into a form that represents its meaning as well as producing a text from the representation of its meaning [94]. A natural language system is, basically, composed of the following components:

- a semantic representation - a knowledge representation scheme (e.g. semantic network).
- a knowledge base composed of syntactic, semantic, and pragmatic knowledge.
- an analysis component to translate a text into a semantic map that represents its meaning.
- a generation component to produce natural language sentences from the knowledge base.

The syntactic knowledge concerns how words can be put together to form sentences that are grammatically correct in the language. This form of knowledge identifies how one word relates (syntactically) to another. The semantic knowledge concerns what words mean and how these meanings combine in sentences to form sentence meanings. The pragmatic knowledge concerns how sentences are used in different contexts and how the context affects the interpretation of the sentence. There is also the morphological knowledge used to analyse how words are constructed out of more basic units called morphemes.

An analysis component translates a text into a semantic map that represents its meaning. The translation process, basically, involves parsing (syntactic processing) , semantic analysis, and pragmatic analysis. Syntactic processing involves the analysis of a sentence and produces a representation of its structure. The syntactic structure is then used as input to the semantic analysis phase which produces an intermediate representation. Pragmatic analysis phase produces a final representation of the sentence.

The generation component works in the opposite direction of the analysis component, i.e. generates natural language sentences from the knowledge base.

4.1.1 Large-Scale Systems

Ideally, a large-scale natural language engineering system should be able to tackle successfully all the separate problems involved in natural natural processing [61]. In engineering terms, a large-scale system has the following characteristics [107]:

- The size of the system (e.g., vocabulary size, grammar coverage) must be sufficient for large-scale applications.
- The system components should be integral parts of the whole. That is, a component should be built so that it can be used effectively to assist other parts of the system without making unreasonable assumptions about those parts.

- The system should be able to adapt easily to various applications in different domains, i.e. flexible.
- The system must be feasible, e.g. hardware requirements must not be too great and the execution speed must be acceptable.
- The system must be maintainable in the sense of any large software system.
- The system must be able to support the applications the users want.
- The system must be robust. This aspect is critical and concerns not only the linguistic scope of the system but also how it deals with input which falls outside of this scope.

Computational linguistic oriented NLP (implementation of a particular linguistic theory/solution to show that it can account for the features it describes) is the other major area in the treatment of NL. Many computational linguistic systems have concentrated on formulating central ideas, but the expansion of these ideas to large-scale systems with the properties listed above has proved a major problem. This is reflected in the small number of systems which can claim to have the properties of a large scale system compared to the abundance of smaller systems which carry out specific tasks in limited domains.

The traditional computational linguistic view that the movement from core ideas to a working system is just a matter of software engineering development seems unfounded as far as NLP is concerned. For example, the execution speed (and thus the feasibility) in a restricted system may be unimportant and only cause problems when the system is expanded. Software development practices might be able to improve the efficiency of the algorithms to some extent but this is unlikely to be sufficient if the complexity of algorithms are high: complexity would not become apparent until the scale of the system is made larger, when no software engineering development could improve the situation significantly (for issues of complexity in large-scale NL algorithms see [75]).

Examples of NL systems are CLARE [9], TEAM, SUNDIAL, KNBL system developed by MCC [18]. CLARE is a similar system to LOLITA (see next section) in the sense that it performs deep

semantic analysis on text in order to arrive at a deep representation of its meaning. CLARE uses a first order logic representation termed *quasi-logical form.*

The KNBL system developed by MCC [18] was developed as an interface to the CYC reasoning database. As this underlying system was not built with NLP in mind, the representation it uses is again different. The size of the grammar and number of rules employed is larger in LOLITA than in both these systems. What is more, in both cases it seems that the NLG is a poor relative whereas the LOLITA generator is treated as a crucial component.

There have been successful methods and formulations which can be classed as generic or general purpose NL generation components as they have been used for different tasks and applications by people other than their creators. Functional unification grammars (FUG) [82] have been used in the generation systems TEXT [82] and TELEGRAM [10]. PENMAN [80] and COMMUNAL [47] both use systemic grammars [92].

Many other NL systems have been built to solve particular problems. They are restricted by the particular task they perform (for example database interface) or by the domain in which they work (for example medical diagnosis).

INTELLECT is a domain independent query system, supplied with basic root vocabulary of 500 common English words. When it detects ambiguities or spelling mistakes, it requests user-assistance. Its grammar analysis is rudimentary: verbs and nouns are not distinguished [106].

4.1.2 The LOLITA System

LOLITA (Large scale, Object-based, Linguistic Interactor, Translator and Analyser) is a general natural language (English) tool under development at the University of Durham for the last 8 years. A block diagram of LOLITA's components is shown in Figure 4.1 [61]. LOLITA is already a large, working system which effectively parses, analyses and generates natural language in an interactive

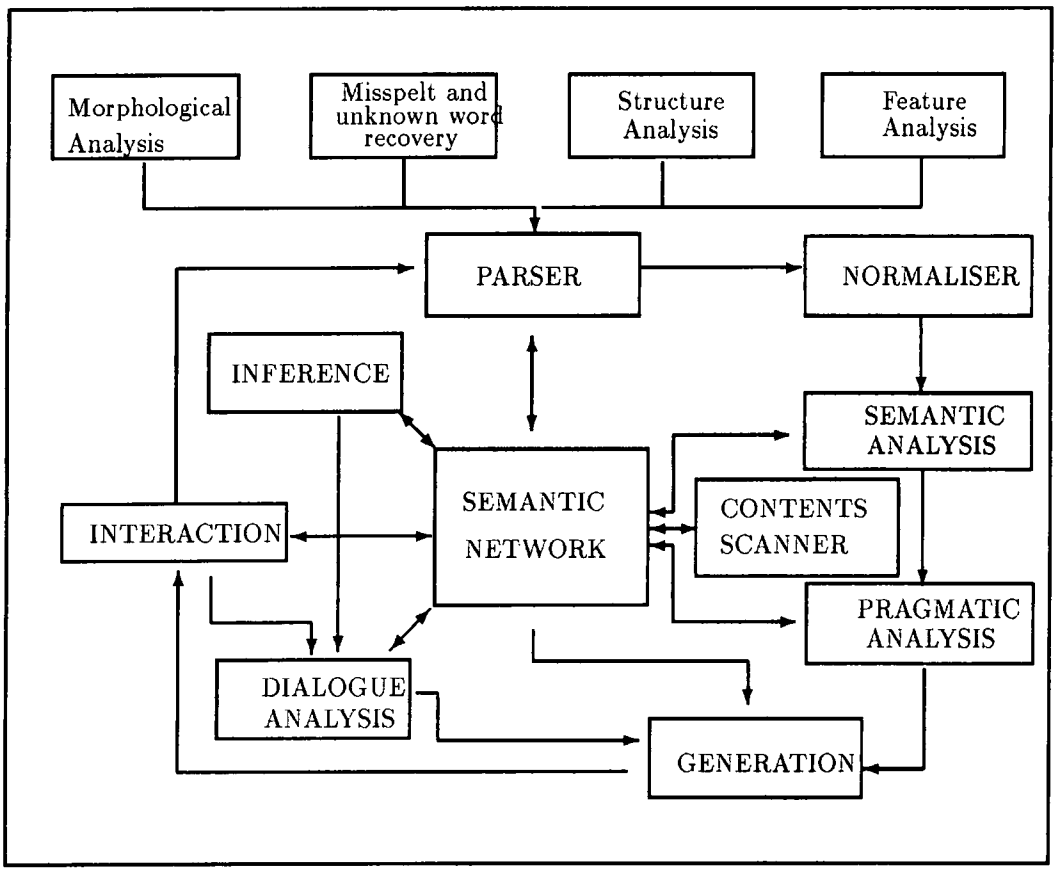

Figure 4.1: Block diagram of LOLITA's components

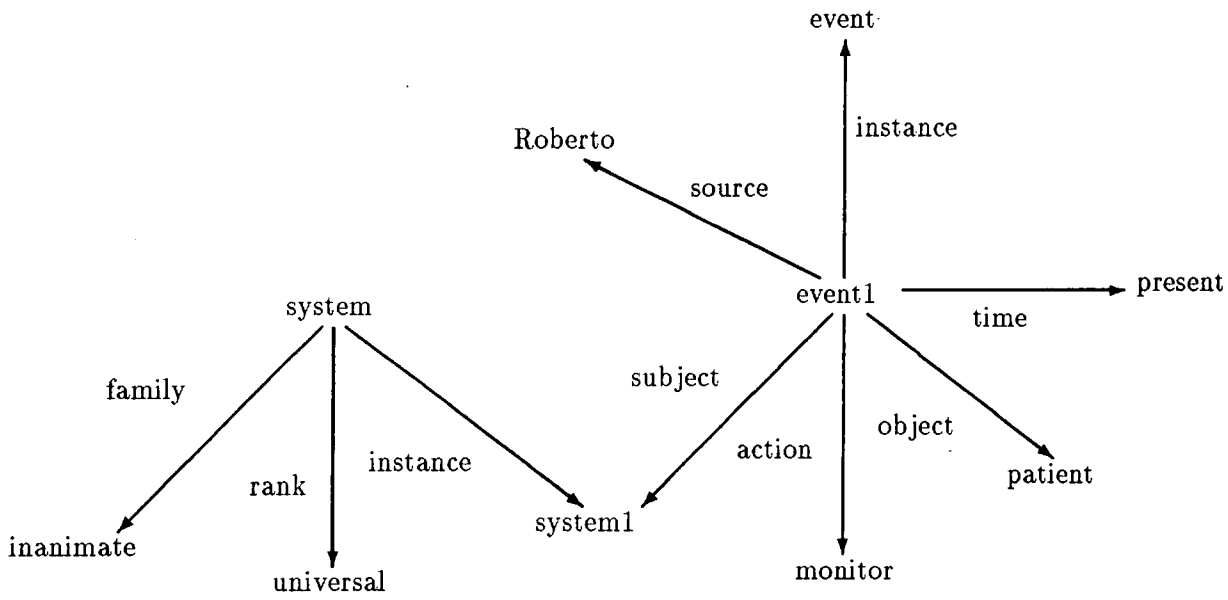

Figure 4.2: Extract from the semantic network

environment. It is modularised, abstracted and very readable allowing various parts of the system to be altered without affecting the rest.

- **The Semantic Network**:- LOLITA is built around a semantic network that holds the system's world information and data as well as some of its linguistic information. The network consists of a set of nodes connected together with arcs. There are three types of entries in the network: entities, relations, and events. Each node is associated with some, from out of about fifty, control variable which holds a rich amount of information about the node. For example, the variable *rank* gives the node's quantification and can have the following values: individual, prototype, general, universal, bounded, named individual, framed universal or class. The semantic network currently comprises 35,000 nodes, with a future expansion to 100,000, capable of over 100,000 inflected word forms. Figure 4.2 is a portion of the semantic network representing the entity 'system' and the event 'the system monitor patients' (event1).

The functions of Lolita can be summarised as follows:

1. **Syntactic Parsing**:- combined with some morphological analysis, parsing produces the best

parse tree or a list of possible parses representing the deep grammatical structure of a piece of text. Morphological analysis produces a surface representation of the text. The punctuations in the input are used to separate it into grammatical units and the spaces to separate it into individual words. Short hand words are replaced by their longer versions (e.g. 'I'll' to 'I Will') and when a word could relate to more than one node in the network, all possibilities are included in the intermediate representation (e.g 'bow' for a ship's bow or a violin bow). Other functions of the morphological analysis include the extraction and the labelling of the roots of the input words. For example morphological analysis on the word 'unworthiness' will extract and label the word 'worth' by separating out the components 'ness' which makes an adjective into a noun, 'un' which indicates a negative. The prepared input representing the surface representation of the text is then ready for parsing where the words and constructs of the natural language input are grouped and labelled into a tree. This is achieved using a 'deterministic grammar and parser' model. A deterministic parser is one that operates on the input in a left to right manner and is able to correctly classify every word and construct every syntactic category once enough information has been accumulated. Each parse tree has all the word features extracted, errors printed out, missing parts inferred and un-parsable parts isolated. LOLITA has over 1500 English grammar rules and is designed to deal with full and serious text such as newspaper articles. LOLITA produces the following parse tree for the sentence:

`The system notifies the emergency services of exceptional health conditions` is:

```
sen
  detph
    det THE
    comnoun SYSTEM [Sing,Neutral,Per3]
  auxphrase_advprepph
    transvp
      comptransv NOTIFY [Pres,Per3]
      detph
        det THE
        snouncl
          adj EMERGENCY [New]
          comnoun SERVICE [Nonum,Neutral,Per3]
    prepp
```

```
prep OF
missing_det
  snouncl
    adj EXCEPTIONAL
    relprepcl
      comnoun CONDITION [Plur,Neutral,NoPer3S]
      prepp
        prep RELATE_
        comnoun HEALTH [Nonum,Neutral,Per3]
```

2. **Semantic Parsing and Pragmatic Analysis**:- maps the deep grammatical representation of the input onto nodes in the network. The semantics must determine whether a node already exists, if and and how to build a new node and how to connect these existing and new portions of the network. An existing node must be identified or a new node built for each object and event involved in the input text. A first step is to make the references absolute. For example, the word 'tomorrow' in the phrase 'I'll do that tomorrow' will be represented by the day after the particular day when this input was analysed. A second stage is to disambiguate the grammatical parse tree in order to decide on one of many interpretations. Pragmatic and more semantic analysis then check that the new portion of the semantic network can fit into the existing network. For example, the sentence "I saw a pig flying" has a correct syntax and its semantic might also be considered well formed. Pragmatic analysis must be able to conclude that there is a problem with the acceptability of this sentence. Alternatively, this sentence may be incorrect under the semantic point of view if there is some definition explicit in the semantic network (via controls) saying that pigs do not fly.

   The following is part of LOLITA's semantic parsing (a number indicates the internal representation of a node) of the event 'the system monitors a patient':

```
* event: 29027 *
 universal_:
   event - 7688 - rank: universal - definition_

 subject_:
   system - 29024 - rank: individual - hypothesis_

 action_:
   monitor - 20384 -
```

```
object_:
  patient - 29026 - rank: individual - hypothesis_

time_:
  present_ - 20989 -

date:
  26 September 1993

source_:
  roberto - 19845 - rank: named individual

status_:
  hypothesis_ - suspended -
```

3. **Generation**:- LOLITA currently produces natural language expressions for nodes in the network, whether they are objects or complex events.

LOLITA meets most of the properties of a large-scale engineering systems [107]:

- **Scale**: As mentioned above LOLITA, operates on a large semantic network and a large number of grammar rules and is able to deal with real-world pieces of text (e.g. newspaper articles)
- **Integration**: The properties of factors such as code length and development time dedicated to the various applications compared with the core LOLITA system is low (e.g. the prototype template filling module consists of about 0.5% of the total code). During development, feedback from specific components affected the development of others (e.g. as the NL generator developed, feedback guided the development of the semantic representation). This procedure is typical of NLE compared to more traditional computational linguistic approaches where isolated development is often practiced.
- **Flexibility**: Although it cannot be used for every task in every domain, LOLITA is not restricted to a single task type (e.g. machine translation) or to a particular domain (e.g. weather reports). This is reflected in the generality of the semantic network representation.

- **Feasibility**: LOLITA runs on a 48Mb Sparc fileserver. Full semantical analysis of, for example, a 100 word paragraph from a newspaper is achieved in few seconds. Generation from the semantic network is in real-time. The algorithms used in LOLITA were specifically designed to have low complexities.

- **Maintainability**: Because the system was built with maintainability in mind the changes made at various stages of the development were easily incorporated.

- **Robustness**: This area is receving a particular attention to increase, for example, the number and coverage of grammatical rules.

- **Usability**: The prototypes which have been already developed are for applications which are in demand from end users although the user interface still need further improvements.

4.2 Natural Language for Requirements Engineering

Whatever technique is used for the requirements elicitation (questionnaires and interviews, work observation, brainstorming, etc), by the far the largest and most important section of available information consists of natural language (NL) statements, often described in terms of what the system shall perform, what functionality the system shall support and what items should exist in the system. It is clear from recent advances of Natural Language Engineering that major inroads are now possible in the interpretation of natural language descriptions. Tools are needed to guide the analyst in the gathering and interpretation of key messages, while keeping track of the interactions with a number of users/experts, during which the analyst's point of view contributes as much as anybody else's to determine the universe of discourse of the problem under discussion.

In recent years a certain number of NLP-based tools have been realised to support the development of Information Systems and the construction of conceptual schemes for Databases. As far as the linguistic theories adopted are concerned, they range from the syntactic linguistic criteria proposed by Abbot [4] to works that attempt to realise a more sophisticated text analysis using variants of

the Case Grammar by Fillmore [53]; however, even in the early work by Abbot the importance of a semantic analysis and of a real-world knowledge is recognised (under the general heading of 'understanding of the problem domain'). The basic approach by Abbot is developed further in a paper by Booch [26] which - together with the one by Abbot - is most quoted with regard to the use of 'linguistic criteria' for the development of models of the external world in terms of interconnected objects.

A process of incremental construction of software modules starting from OO specifications obtained from informal NL specification is described in [105], in which the transformation process takes place interactively and the analyst's role is very important, since the system is able to extract verbs and nouns from an informal specification automatically, but not to determine which words are important for the construction of a formal specification. With respect to the approach by Abbot, the role of verbal patterns are underlined. The NL specifications are analysed separately using two dictionaries, one for nouns and another for verbs, which are classified in a way related to the OO model used for formal specifications. The authors accept that the correspondence between the lexical, and semantical, structures of words and the structure of software components does not always hold. Thus, attention should be paid to the semantical roles of nouns and verbs. Furthermore, the rules for extracting information from the informal description are ad-hoc and are specific to an object-oriented model.

In order to limit this problem, Bailin [13] proposes a different solution, based on 'filtering' the requirements text to build a requirements Database, so as to simplify the search for nouns and noun-phrases. However, no details are given on how to achieve this simplification. LOLITA can constitutes a strong support tool in the initial phase thanks to its 'text normalisation' capabilities. Another significant aspect in Balini's work is the proposal to use Data Flow Diagram (DFDs) in order to identify entities in the problem domain (although the author maintains that DFDs are not really part of OO specifications, and can only be used as an intermediate representation).

A well-known method for the construction of conceptual schemas for relational Database development is NIAM [44]. In order to obtain a domain-independent system, NIAM uses the result

of the context-based NLP disambiguation. The analyst interacts with the system on a particular universe of discourse, utilising a subset of the NL which is similar to that produced by the LOLITA normaliser: just like other similar NLP approaches, this one lacks in generality, as it was built with that particular application in mind.

The Requirements Apprentice (RA) [99] is among the systems devoted to the early stages development support for systems based on AI-style knowledge acquisition. RA makes an extensive use of domain knowledge in the form of clichés and requires input from an analyst to assist resolving ambiguities and contradictions, and filling in details. To avoid the problems related to the complexity of the free language which the user is likely to employ, RA imposes a more restricted command language: this is an approach that may help the analyst, but not the requirement provider.

The expert system ALECSI-OICSI [33] allows the user to express the requirements both in NL (French) and in graphic form, and the system builds conceptual structures. ALECSI uses a semantic network to represent the domain knowledge. The NL interpretation is based on the case semantic approach [53], according to which the meaning of a sentence can be extracted from the meaning of the verb and the recognition of the connected cases. By applying the 'case' concept to the sentence level too, it is possible to obtain a top-down approach to the interpretation of complex phrases. Furthermore, since the linguistic patterns are independent from any particular model building technique, they can be used with any development methodology.

OICSI is based upon the REMORA methodology, which identifies four basic concepts: objects, actions, events, and constraints, and four corresponding types of nodes in the semantic net used to implement the conceptual scheme needed. There are five types of arcs, representing respectively: a relationship between two objects, an action modifying an object, an event triggering an action, an object changing state, and a constraint on a node.

An extension of the case grammar is used also in the requirements analysis support described in [20]. The prototype automatically performs a 'semantic case analysis' on a sentence based on a subset of English (Analyst Constrained Language: ACL). It uses a form of Predicate Calculus, rather

than a semantic net formalism. Its system architecture is based on four modules: three of which constitute the NLP system, while the fourth is the Object-Oriented Analysis System (OOAS). The OOAS takes as input from the NLP modules an annotated tree and case frames. From these two structures, roles, relationships and inheritance rules are derived, which are represented in relational tables. These tables are then used for requirements elicitation. In order to identify which noun phrases from the input text represent candidate objects, heuristics are proposed for the analyst to apply. It is also claimed that, as a by-product, the prototype allows the automatic identification of ambiguities, inconsistencies, etc.

Compared with ALECSI-OICSI system, this is a less sophisticated system: rather than use a robust NLP system, it restricts the range of possible input texts, and takes advantage of the fact that the style in technical documents is simpler.

Among other support tools for the reorganisation and normalisation of NL input is the Fact Gathering and Analysis Tool [101]. which aims to assist the analyst in collecting and digesting facts from end-users. This is, however, only a tool for the orderly classification of documents for automatic controls (such as aliases, homonyms, etc.).

In Easterbrook's model [45] interview transcripts (from dialogue between analyst and the client) are interpreted into some formal language (first order predicate calculus). The process starts with breaking the textual information into chunks, where each chunk focuses on a particular area of knowledge (or a topic). Each chunk is identified by its source, where a source could be a person or a group of people, and then interpreted into a set of propositions (statements) which act as a formal representation of the information contained in that chunk. For example, the two propositions:

```
goal(max_access)
```

and

```
goal(max_access) -> goal(max_books_on_shelves)
```

is an interpretation of the following piece of dialogue:

```
Librarian:  The main aim of the library is to
```

maximise access to books.

Analyst: Okay. How do you measure success?

Librarian: Well really it boils down to maximising the number of books on the shelves.

The model concentrates on the analysis of questionnaire responses rather than on how to obtain them and relies on the analyst's skill at interpretation. In addition, the model does not allow any automated reasoning with the information it captures.

4.3 Application of LOLITA to Requirement Engineering

LOLITA can intervene in the initial phases of requirements engineering by analysing input texts at several levels in order to correct, select and normalise them, so as to generate a problem statement which in turn would be the starting point for requirements modelling. There are functionalities in LOLITA which allow text correction, and completion, style differences elimination (eventually even differences in the actual language used), ambiguity resolution, (at the grammatical, semantic, pragmatical, discourse and dialogue levels): all these are phenomena which occur very frequently in real and serious text. They are also features that require a real general Natural Language system, which is why so many other problem specific systems use a pseudo-natural language.

Real NL text also brings problems of redundancy, inconsistency and omission management: as already pointed out, these are not problems connected to the automation of the process: they are intimately connected to the user-analyst relationship, in the requirements elicitation process itself, and have to be faced no matter what method, tool or system is used. Furthermore, the possibility of interacting with the user through 'query' and 'dialogue' modules, and of generating pieces of text allows the requirements elicitation to be run interactively via a NL interface. Finally, it is also possible to interact with the system with a graphical interface (developed by Siemens Plessey), which allow exploration and manipulation of the semantic net (see figure 4.3). Once redundancies

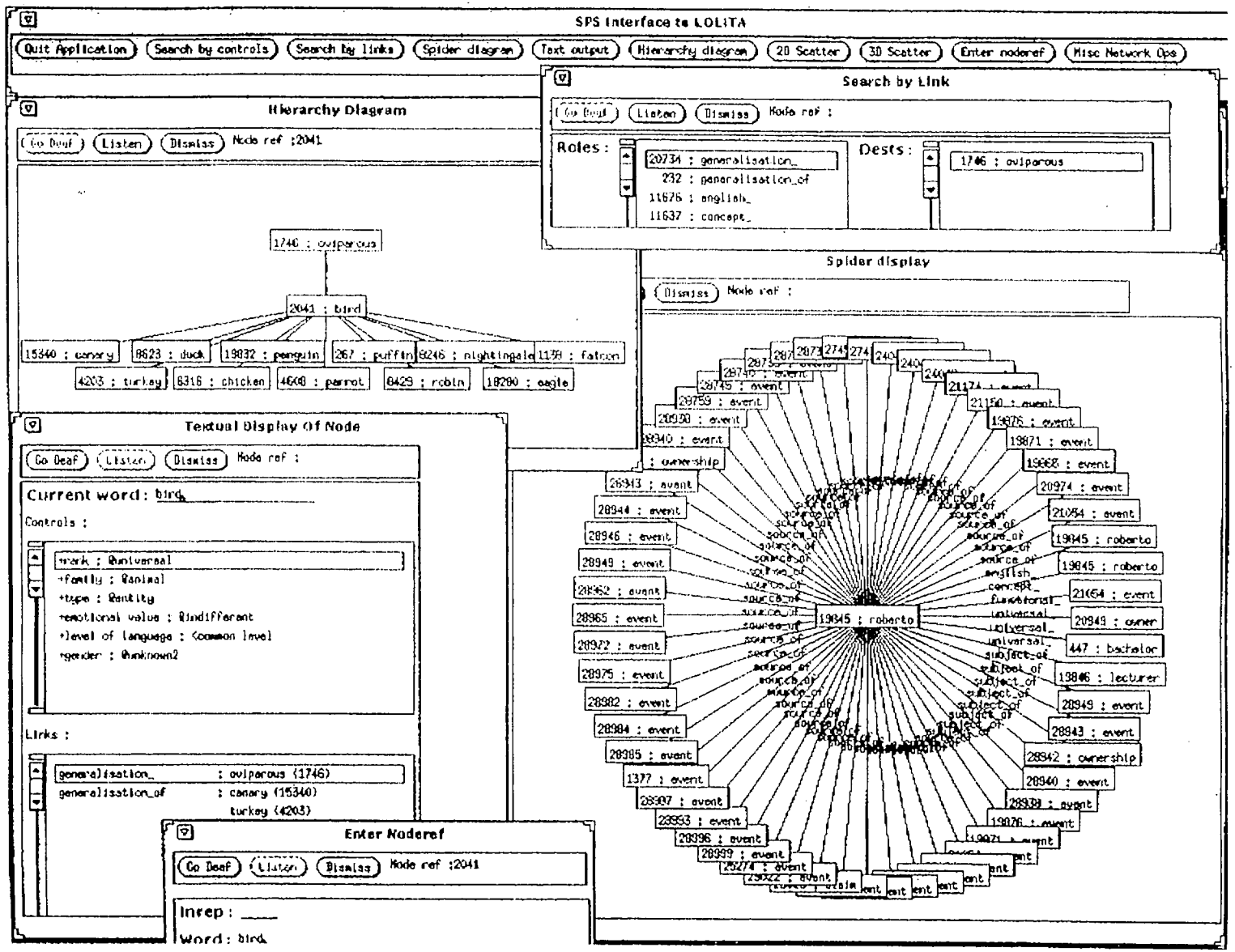
SPS Interface to LOLITA
Quit Application
Search by controls
Search by links
Spider diagram
Text output
Hierarchy diagram
2D Scatter
3D Scatter
Enter noderef
Misc Network Ops
Hierarchy Diagram
Go Deaf
Listen
Dismiss
Node ref :2041
1746 : ovipar ous
2041 : bird
15340 : canary
8623 : duck
19832 : penguin
267 : puffin
8246 : nightingale
1139 : falcon
4203 : turkey
8316 : chicken
4608 : parrot
8429 : robin
18280 : eagle
Search by Link
Go Deaf
Listen
Dismiss
Node ref :
Roles :
20734 : generalisation_
232 : generalisation_of
11676 : english_
11637 : concept_
Dests :
1746 : ovipar ous
Spider display
Dismiss
Node ref :
19845 : roberto
Textual Display Of Node
Go Deaf
Listen
Dismiss
Node ref :
Current word: bird
Controls :
rank : Universal
+family : Animal
+type : Entity
+emotional value : Indifferent
+level of language : Common level
+gender : Unknown2
Links :
generalisation_ : ovipar ous (1746)
generalisation_of : canary (15340)
turkey (4203)
Enter Noderef
Go Deaf
Listen
Dismiss
Node ref :2041
Inrep :
Word: bird

Figure 4.3

and inconsistencies in the semantic net, have been resolved or flagged for user's attention, conceptual modelling can begin using a particular modeling technique (e.g. Object-Oriented).

Other applications, outside the software engineering field, are under considerations. These include:

- Context scanning for security operations.
- NL language interface for SQL (DEAC project, funded by ESPRC).
- Dialogue training for educational purposes.
- Chinese language tutoring.

4.4 Summary

In this chapter the case has been argued for real NLP systems as a tool for requirements engineering. Various approaches have been discussed, and the fundamental common drawbacks pointed out: these may manifest themselves as the need for a pseudo-language, or heavy domain dependency, or excessive reliance on user interaction, but the drawbacks are due to the lack of proper NL facilities (at the analysis, reasoning, and generation stages). The Natural Language Engineering paradigm has been briefly presented, and a claim made that is the way forward for real NLP. The LOLITA system has been presented, in relation to this paradigm, and shown to have the range of functionality needed for many aspects of requirements engineering, requirements elicitation in particular.

The next two chapters address the specific problem to be solved in this thesis. To restate the problem: develop a requirements validation method that enables the very early detection of problems in the requirements, and their subsequent resolution. The developed method should improve on the Viewpoint Analysis method proposed by Leite. That is, the method should detect better quality problems, detect problems earlier than the Leite's method allows and should support the nego-

tiation process for resolving those problems. To remind the reader, Leite uses an agent-oriented approach to viewpoint resolution as a means for early validation in the process of requirements elicitation.

Chapter five lays down the principles of an alternative, agent-oriented approach to viewpoint resolution. The chapter describes how these principles were established. Chapter six describes a validation method that uses the viewpoint resolution principles, established in chapter five, as a basis. The idea behind presenting viewpoint resolution and requirements validation separately is to illustrate the fact that in order to improve on the validation method proposed by Leite his viewpoint resolution strategy should be improved. The improvement concentrates on the explicit use of human factors and relations in order to deal with the uncertainty in the information coming from different viewpoints.

Chapter 5

Viewpoint Resolution Through Source Control

As shown in Chapter 2 a viewpoint resolution method is a process of identifying viewpoints, reasoning within a viewpoint, reasoning between different viewpoints, and revising a viewpoint. This chapter describes an approach to viewpoint resolution. The basic principles of the approach stem from combining ideas from the fields of uncertainty management and requirements engineering. The strategy used to establish these principles consists of mapping an established AI model onto the domain of requirements engineering. The Source Control Mechanism (SCM) is selected as an AI model for the management of uncertain information from multiple human sources. Section 5.1 introduces the mapping strategy. Section 5.2 gives an overview of the Source Control Mechanism. Section 5.3 describes the adaptation of the SCM to the requirements engineering domain using the mapping strategy introduced in section 5.1. Finally, an outline of the viewpoint resolution principles is given followed by a summary.

5.1 A Mapping Strategy

In a paper entitled *a framework for incremental progress in the application of Artificial Intelligence to Software Engineering*, Freeman and his colleagues [12] propose a framework designed to uncover opportunities for incremental progress in SE practice through adoption of AI solutions. The aim of the technique is to decompose the SE-AI universe into manageable pieces called *local environments* - subspaces resulting from the intersection of the different perspectives of the universe.

A local environment defines a particular SE problem such as specifications evolution and an AI "resource" such as current truth-maintenance technology.

The interactions within a given local environment are then studied using the modeling approach to assist in mapping a given domain from AI onto a related domain from SE, in order to identify possible contributions of AI "resources" to SE problems. In essence, the modeling process is an approach in which the study of a domain D (for which no theory is known) is replaced by the study of some other set of facts organised into a model M for which a theory is known, and that has important characteristics in common with the domain under investigation. The aim is to construct a model that better mirrors a given domain enabling it to be unambiguously understood and exposes its limitations.

Thus, given an AI model M and a SE domain D which exhibit interesting commonalities, and the model M seems to have a better developed understanding (representation, technique, tool, etc.), the modeling process involves:

1. Defining a correspondence between D and M,
2. Analyzing the structure or the behaviour in M, and
3. Transferring the conclusions of the analysis back to D and validating it in the context of D.

The paper reports that only step 1 has been addressed. The step is called "term correspondence".

Once a restricted local environment has been defined, term correspondence involves the following procedure:

1. List the concepts involved in the definition of the SE problem. For each concept, list its associated properties and relations to other concepts.
2. Do the same for the concepts involved in the AI resource.
3. Establish a correspondence between SE terms and AI terms which exhibit similar properties or relations.
4. Complete a modeling map between the SE domain and the AI domain.

Our restricted local environment is composed of the Source Control Mechanism and the requirements elicitation domain. Using this framework the concepts of the Source Control Mechanism are mapped onto the concepts of requirements elicitation. The mapping is presented in the form of 'levels'. At each level a SCM concept and its image in requirements elicitation domain are described, followed by a justification of the adaptation. Since the SCM is heuristic-based most of the concepts, from both domains are described as sets of heuristics.

5.2 Truth Maintenance

The main idea of the work reported in this thesis is to view requirements elicitation as a truth maintenance process.

The basic idea behind truth maintenance is belief revision [41]. Many problems involve the need to make a decision about how to proceed with less than perfect information and truth maintenance provides a way of allowing the consequences of assumptions to be determined and the set of beliefs revised, if necessary. Truth Maintenance Systems (TMSs) are house-keeping sub-systems of reasoning systems. The problem solver passes the inferences it makes to the TMS, which in turn organises

the beliefs of the problem solver. As the TMS has no access to the semantics of the information, it is usually the responsibility of the problem solver to ensure correctness of information.

In general, the TMS provides two services to the problem solver, truth maintenance and dependency directed backtracking. Truth maintenance is required when a piece of information which is currently disbelived is given a valid justification. The TMS must therefore calculate the status of other pieces of information which are in any way connected or affected by the change. Dependency Directed Backtracking is required when a piece of information which has been declared as valid is found to cause a contradiction. The TMS finds the ste of assumptions on which the justification for the contradictory information depends. One of the assumptions from this set (culprit) is retracted. If this does not succeed in forcing the contradiction out, the process is repeated.

5.3 An Overview of The Source Control Mechanism

The Source Control Mechanism [24] is a model for the management of uncertain information, from human sources, through belief revision. The main principle of the Source Control Mechanism is that human agents tend to build models about other human agents they are acquainted with. These source models record factors concerning their opinion about the abilities and trustworthiness of individual sources of information. The source models are used to evaluate information received from the respective sources. They are then reassessed in the light of feedback from the results of the process of information evaluation and belief formation.

A source model keeps a record of a source's performance in providing information. It represents the system's opinion about the source and encapsulates expectations on the source's behaviour. The SCM attempts to learn about the behaviour of its environment with the view to preempt and anticipate situations that carry the potential of serious contradictions.

The SCM operates on two levels: an object-level and a control-level. At the object level pieces of information are processed until a contradiction occurs. The control level attempts to resolve the

contradictions and, at the same time, uses the contradictions as a signal to trigger an evaluation of the information sources.

The object-level involves:

- Defaults and Classification. Default source models for new sources are created using a default and classification mechanism. This is based on the observation that human agents frequently have to evaluate information from sources they do not know much about. In the absence of concrete evidence that can be gained from actual experience with the source or reported by third parties, human agents tend to quickly create a source model based on some class to which the source can be associated with and whose properties are known or, as a last resort, use defaults values.

- Importance Analysis. Importance Analysis is Approximating the amount of effort that needs to be invested in the analysis of a piece of information. If there is a problem with a piece of information Importance Analysis decides how far to go into investigating the problem.

- Information Evaluation. As the information from human sources is largely of uncertain nature, human agents have to decide whether, or how much, to believe individual pieces of information. The SCM uses the external features of a piece of information (such as source, tone, language, consistency, etc) rather than the content of the information itself.

The control-level mechanism is based on the following:

- Conflict Resolution. Conflict resolution heuristics use information from the source models to sort out contradictions between information, from the same source, and from different sources. For example, the opinion of the most reliable source prevails.

- The Principle of Enquiry. An enquiry is launched if more information is needed. Information may be required to find out more about a particular source or for more evidence to support or weaken an existing piece of information.

- Source re-evaluation. Information about a source, recorded in the source model, needs to be adjusted in the light of new evidence about the source's ability and trustworthiness, that may emerge from the analyses and enquiries.

In order to form beliefs about a given information, the SCM uses a collection of general heuristics to extract the various parameters from that information and to make a decision about it, on the basis of those parameters.

The Source Control Mechanism is well suited for applications of investigative nature such as requirements elicitation.

5.4 Adaptation of The Source Control Mechanism

One of contributions of this work is the application of the truth maintenance techinque to requirements elicitation. The Source Control Mechanism is an example of a system that uses truth maintenance technology as a basis.

Although the Source Control Mechanism is domain-independent, it can be applied more successfully to domains where there is a continuous flow of information from human sources operating in the same domain, and where precision is of less importance than establishing evidence about the situation under analysis.

This section describes how the SCM is adapted to the requirements elicitation domain using the mapping strategy, described in section 5.1., at each level of abstraction. At each level of abstraction we give the concepts of the SCM and their adaptation to requirements elicitation, followed by a justification of the adaptation.

To remind the reader, the Source Control Mechanism is a model for the management of uncertain information, from human sources, through belief formation. The main principle of the Source

Control Mechanism is that human agents tend to build models about other human agents they are acquainted with. These source models record factors concerning their opinion about the abilities and trustworthiness of individual sources of information. The source models are used to evaluate information received from the respective sources. They are then reassessed in the light of feedback from the results of the process of information evaluation and belief formation.

A source model keeps a record of a source's performance in providing information. It represents the system's opinion about the source and encapsulates expectations on the source's behaviour. The SCM attempts to learn about the behaviour of its environment with the view to preempt and anticipate situations that carry the potential of serious contradictions.

The SCM operates on two levels: an object-level and a control-level. At the object level pieces of information are processed until a contradiction occurs. The control level attempts to resolve the contradictions and, at the same time, uses the contradictions as a signal to trigger an evaluation of the information sources.

The object-level involves:

- Defaults and Classification. Default source models for new sources are created using a default and classification mechanism. This is based on the observation that human agents frequently have to evaluate information from sources they do not know much about. In the absence of concrete evidence that can be gained from actual experience with the source or reported by third parties, human agents tend to quickly create a source model based on some class to which the source can be associated with and whose properties are known or, as a last resort, use defaults values.

- Importance Analysis. Importance Analysis is Approximating the amount of effort that needs to be invested in the analysis of a piece of information. If there is a problem with a piece of information Importance Analysis decides how far to go into investigating the problem.

- Information Evaluation. As the information from human sources is largely of uncertain na-

ture, human agents have to decide whether, or how much, to believe individual pieces of information. The SCM uses the external features of a piece of information (such as source, tone, language, consistency, etc) rather than the content of the information itself.

The control-level mechanism is based on the following:

- Conflict Resolution. Conflict resolution heuristics use information from the source models to sort out contradictions between information, from the same source, and from different sources. For example, the opinion of the most reliable source prevails.
- The Principle of Enquiry. An enquiry is launched if more information is needed. Information may be required to find out more about a particular source or for more evidence to support or weaken an existing piece of information.
- Source re-evaluation. Information about a source, recorded in the source model, needs to be adjusted in the light of new evidence about the source's ability and trustworthiness, that may emerge from the analyses and enquiries.

In order to form beliefs about a given information, the SCM uses a collection of general heuristics to extract the various parameters from that information and to make a decision about it, on the basis of those parameters.

The following are the different levels of abstraction at which the SCM techniques are adapted to the requirements elicitation domain, starting with the concept: uncertainty management.

5.4.1 Level 1

Uncertainty management The fundamental principle of the Source Control Mechanism was born out of the need to manage the uncertainty of the information that one gets from human sources in order to make a sense of a particular subject matter. Its strategy is to form beliefs

about the information using its view about the sources and then modifies those beliefs in the light of what it has learned about the sources.

Requirements Elicitation Uncertainty in computer applications is certain and software engineering is an attempt to manage that uncertainty [71]. This is particularly true for requirements elicitation from multiple human sources (see Chapter 1).

Justification Requirements engineering is a people-oriented job. The use of multiple viewpoints in requirements elicitation is akin to a court investigation where different witnesses may have conflicting or corroborating views [72, 51]. This is also the principle used by the SCM.

5.4.2 Level 2

Initial source models For each new source establish an initial source model using a default and classification mechanism in the absence of concrete evidence about the actual properties of the source. A source model for an engineering configuration manager is represented by the SCM as:

```
Manager
   Ability:
      expertise: engineering_configuration_management
      experience: high
      reasoning : high
      interests : improve the quality of the control construction

  manager ----> analyst
      helpfulness : average (default)
      trustworthiness : high

 analyst ----> manager
```

```
helpfulness : high
trustworthiness : high
```

Requirements Elicitation Select an initial set of viewpoints that would take part in the viewpoint resolution process using some form of pruning mechanism and then create a viewpoint hierarchy. Examples of heuristics for constructing a valid viewpoint hierarchy were recommended by the CORE method [88]:

```
if a viewpoint has more than one responsibility
  or is responsible to several superiors
  then it should be re-examined
```

Establish the initial models of the selected viewpoints using their area of responsibility, area of expertise and their experience. In the context of requirements elicitation the 'interests' parameter will be interpreted as goals the source wants to achieve. Other viewpoints are included as the investigation progresses.

Justification Requirements elicitation requires a context (universe of discourse) in which the information will be assessed. Fickas [51], for example uses domain goals as a universe of discourse to validate specifications. The Universe of discourse in this thesis is composed of the viewpoint models that capture records of the information sources. A viewpoint model will play a crucial role in requirements elicitation:

- it captures a detailed track record for individual agents.
- it ties pieces of information to the universe of discourse
- it will be used to assess information
- it will be used in case of negotiation required to resolve conflicts
- it can be used in other activities of the software development process if the corresponding agent is involved (e.g., during system maintenance)

5.4.3 Level 3

Importance analysis Given a source model and piece of information there are three ways in which importance can initially be established:

```
if one has an interest in the subject of the information,
      then the information is important

if the information is strong and there is a connection to existing,
important information then the information is important

if one's helpfulness towards the source is high,
     then the source is important

if the source is trustworthy and competent
     then the source is important
```

Other heuristics are concerned with situations where a piece of information which has been analysed already and where the importance analysis has to decide whether there is enough interest in the situation to make further enquiries. For example:

```
if there is a problem and the information and the source are
of interest,  then recommend not to enquire

if there is a potential problem with ability
  and the source is important to the mechanism,
   then recommend source analysis and enquiry
```

Requirements Elicitation The initial assessment of relevance could be guided by the following heuristics:

```
If the information lies directly within the viewpoint's
```

```
responsibility, knowledge, and experience
     then the information is relevant

If the analyst has knowledge of the application domain
   then use that knowledge to make the best approximation
    of relevance

if the viewpoint's representative is important in the
   organisation's hierarchy
    then the information is relevant

if the analyst has an interest in the subject of the information,
        then the information is relevant
```

Justification The following case illustrates the importance of the assessment of relevance (quoted from [76]):

> *a user gave a very comprehensive account of one subject which was officially his responsibility and implied he had knowledge of it. It turned out later, much later, that the man had only been in the job for less than three weeks. His answers were theoretical ones; how he thought logically it should be done.*

5.4.4 Level 4

Information Evaluation Information evaluation is concerned with assessing the credibility of a piece of information both from its well-formedness stand-point and against the *source model.* Information evaluation operates both on the properties of the information and the properties of its source. At the information level the following parameters are involved:

- the relative strength of the argument
- whether responsibility is accepted

- whether the source has an interest in including the information

At the source level, the ability and trustworthiness of a source are considered, namely:

- expertise - includes subjects that the source has had particular training in, as well as expertise in general, common knowledge and more practical experience,
- experience - the source's ability to judge and handle correctly its own, personal experiences,
- reasoning - the source's ability to correctly follow and handle long chains of reasoning. It is not limited to particular areas of the source's expertise,
- interests and beliefs,
- judging information - the source's ability to handle and evaluate information it receives from other sources.

Requirements Elicitation Given a viewpoint model the following is an example of requirements evaluation heuristics:

```
if the conviction in the information is high
   and the viewpoint  denies responsibility
   and the trustworthiness of the viewpoint is low
   then  record trust problem
   and reject the information
elseif the agent is trustworthy
  then record ability problem
     and modify belief according to the ability

if the conviction in the information is low
    and the viewpoint  denies advantage
    and the trustworthiness of the viewpoint is high
    and the ability is high
  then record o.k. and accept information as given
```

Justification Analysts, often encounter conflicting interests some of which may be hostile to successful operation of the proposed system. An analyst must anticipate views that accidentally or intentionally, might lead to unacceptable situations. It is necessary, therefore, where possible, that opportunities for those views must be eliminated [8]. Many authors recognise the role of human factors in requirements analysis but no one treats those factors explicitly. Fickas *et al.* [52] suggests a set of heuristics for defining which system human agents should best perform which actions. Agents are assigned to actions depending on their ability, reliability and motivation. For example, no agent will be responsible for a goal in conflict with its private goal or if there are several candidate agents to perform an action an agent is selected so that the values of the ability and reliability are maximised. Mullery and Finkelstien promotes the idea of commitments, that is to hold people accountable for statements they make or decision they take, in order to encourage responsible attitudes.

5.4.5 Level 5

Conflict Analysis Single source and multiple sources conflicts are considered by the SCM. Four types of conflicts are defined:

- Reiteration
- Weakening
- strengthening
- Contradiction

The principles of single-source conflict analysis (i.e. information from the same source) are the same for multi-source conflict analysis. As in information evaluation, conflict analysis heuristics use the external features of the information. The SCM does not provide a method for detecting conflicts, except few heuristics for contradiction analysis. For example, given a piece of information:

```
if the source accepts responsibility
```

```
and the convictions for and against are of the same strength
 then there is a contradiction
```

Requirements Elicitation In this section, only multi-viewpoint conflict analysis heuristics are considered. Single-viewpoint conflict analysis heuristics follow the same principles with slight variations.

Assuming that a change of environment is not plausible, part of the decision tree looks like:

In the case of reiteration

```
if the old conviction is roughly equal to the new
   and  the levels of conviction are high
   and there is no problem of trust
   and there is problem of ability,
        then add viewpoint and checkout ability problem

if if the convictions are equally high
   and there is no problem of ability,
        then add viewpoint
```

In the case strengthening:

```
if the new information is weaker than the old,
   and investigation reveals that there is some substance
      then keep old belief and add viewpoint,
      otherwise keep old belief and do not add viewpoint

if the new information is stronger than the old,
   and there is a problem of ability,
     then raise to level of ability and add viewpoint

if the new information is stronger than the old,
```

```
    and there is no problem of ability,
      then raise to level indicated and add viewpoint

  if the old information is corroborated,
    and the new information is stronger than the old,
      then raise belief and add source

  if the old information is corroborated,
    and the new information is equally solid,
       then raise belief and add source

  if the old information is corroborated,
    and the new information is less solid,
       then keep belief at present level and may be add source
```

In the case weakening:

```
  if the new information is weaker than the old,
       then keep old belief and add viewpoint,
       otherwise keep old belief and do not add viewpoint

  if the new information is stronger than the old,
    and there is no problem of ability,
       then reduce belief and add source

  if the old information is corroborated,
    and the new information is equally solid,
       then marginally reduce belief and add viewpoint

  if the old information is corroborated,
```

```
    and the new information is less solid,
      then keep belief at present level and may be add source
```

and in the case a contradiction:

```
  if the new information is weaker than the old,
     then keep old belief and add source

  if the new information is stronger than the old,
     then suspend and investigate

  if the two positions are equally solid,
     then keep relative weight of beliefs, and add new viewpoint on
        opposing side

  if the old information is corroborated,
     and the new information is equally solid,
      then marginally reduce belief and add viewpoint
         and supports on opposing side at most at average level

  if the old information is corroborated,
     and the new information is less solid,
      then keep belief at present level and may be add source
         on opposing side
```

Justification Conflicts analysis is a crucial part of a multi-viewpoints method and should be treated as an explicit activity. Having adopted the principle of a court investigation as a basis for requirements elicitation it follows that the four situations: reiteration, weakening, strengthening, and contradiction apply in requirements elicitation. Conflicts are used here as trigger to uncover further information and to learn more about the problem under investigation. There is no consensus over what is a conflict and what is not. Each author adopts their

own definition. In this this thesis a 'judicial' approach adopted as advocated by the Source Control Mechanism (more details are given in section.

5.4.6 Level 6

The principle of Enquiry An enquiry is launched if :

- there is a need for further information necessary to find a solution to a question
- there is a need for more evidence to support or weaken an existing piece of information
- there is a need to find out more about a particular source

The SCM first decides whether it is worth launching an enquiry by using a different type of importance analysis. For example:

```
if there is a problem and the information and the
     source are not of interest,
     then recommend not to enquire

  if there is a problem and the efforts required to solve it
     is greater than the source or information warrants it,
     then recommend not to enquire

  if there is a problem with ability
     and the source is of interest,
     then recommend source analysis an enquiry

  if there is a problem with ability
     and the information is of interest,
     then recommend information analysis an enquiry
```

```
if there is a problem with trustworthiness
   and the information is of interest,
   then recommend an enquiry

if there is a problem with trustworthiness
   and the source is of interest,
   then recommend source re-evaluation and investigation

if there is a conflict with the other information
   and the source or information is of interest,
   then recommend further investigation
```

Requirements Elicitation Further to the above heuristics there are other domain-specific heuristics. for example:

```
The analyst should encourage viewpoints to
volunteer information

The analyst should make the maximum use
of the information available, e.g. use the
properties of the requirements language to infer other
information
```

Justification Requirements elicitation is an investigative process of exploration and learning. Collecting the maximum information in the minimum of time requires maintaining a balance between interaction with the information sources and making the maximum use of information available.

5.4.7 Level 7

Source Re-evaluation The SCM revises the indices in the model of an existing source in the light of new evidence or creates a new model for a new source based on the available information about that source. It must change:

- the ability related indices - eg. expertise, reasoning, etc
- the trust related indices - eg. beliefs, interests, special relationships

To change the ability index the SCM employs the following heuristics:

```
if there is a new case,
   and there is no connection to other information
   and there is no specific evidence,
     then add the type of case to the accumulated evidence

if there is a regular pattern in the records,
      then check whether that pattern can be
        explained in the source model

if there is a pattern which cannot be explained
    by the source model,  and the index is built on long
    standing evidence,
       then keep accumulating evidence and investigate
    elseif the index is not built on long-standing evidence,
        then weight index against evidence and adjust index

if the source makes strong technical claims
    and the index does not record any expertise,
       then investigate whether there could be a classification
       to explain it
```

else enquire with the source or sources who would know

if there is an opportunity to talk to a source,
then ask about its schooling, training and profession
and analyse the types of abilities required for that

if there has been an enquiry into the source,
and there is evidence of deficiencies or abilities
and the abilities are not reflected in the index,
and the evidence is stronger than the index,
then adjust the index
elseif the evidence is not stronger than the index,
then add evidence to records

if a source is reporting about another source
and the reporting source is good at judging sources
and there is no problem of trust,
and the report is stronger than the index,
then adjust the index
elseif the report is not stronger than the index,
than add evidence to records

if a source is reporting about another source
and the reporting source is good at judging sources
and there is problem of trust,
then investigate further or discard evidence

if there is evidence of classification,
and classification can be explained by past performance,

```
        then apply classification
      elseif there is no past performance to judge against,
        then apply classification
      else investigate and record evidence
```

To maintain the belief index:

```
    if there is a regular pattern in the records,
      and that pattern is about opinions
      and the source keeps reiterating its opinion
      and there is direct evidence,
          then add belief to list
      elseif there is no direct evidence,
        then record possibility of a strong belief

    if there is evidence about education and social situation,
      and that has strong beliefs associated with it,
       then add beliefs to the source model

    if there is an association with a particular class,
      and that class has strong beliefs associated with it,
       then add beliefs to the source model

    if there is a belief and the source does not behave
      in accordance with it, and
      there is no previous evidence for that belief,
       then remove belief
      elseif there is previous evidence,
       then investigate and add to records
```

```
if is an opportunity to talk to a source,
   then ask about its schooling and profession
   and analyse the types of classes which may apply
```

For the maintenance of the interest index:

```
if there is a regular pattern in the records,
   and that pattern is about making strong claims
   while denying responsibility
   and there is a connecting factor which implies
   some form of gain,
   and there is direct evidence,
    then add interest to list
   elseif there is no direct evidence,
    then investigate or record possibility of a strong interest

if there is an association with a particular class,
   and that class has strong interests associated with it,
     then add interests to the source model

if there is an interest and the source does not behave
   in accordance with it, and
   the interest comes from a classification,
      then remove interest
   elseif there is previous evidence,
    then investigate the classification

if there are strong relationships involved,
   and there is a pattern of similar behaviour,
   then record interest for when these relationships
```

```
        and the subjects are involved
```

To maintain the trustworthiness index:

```
    if there is no record of a breach of trust in the records,
       and the number of recorded instances are significant,
        then general trustworthiness --> high
       elseif there are minor problems with trust,
        then general trustworthiness --> average
       else if there is a sudden problem with trust,
       and the source is important,
        then investigate or record unresolved problem of trust

    if a significant record of helpfulness,
       and there is no obvious fracture,
        general helpfulness --> high
     elseif there is no consistent record,
       then general helpfulness --> average
     else if there is a consistent record of being uncooperative,
       then general helpfulness --> low

    if there is a pattern of helpfulness or trustworthiness
       associated with a particular subject,
       and there is a strong interests associated with the subject,
       and there is a relation associated with that interest,
          then create that relation for the source model
          and calculate trustworthiness and helpfulness from records

    if there are strong relationships associated with the source,
       and they are part of a class which has strong interests
```

```
associated with it,
  then add new relation and abstract typical
   behaviour from a semantic definition
```

Requirements Elicitation As above.

Justification As pointed out by Fickas [51] we get a radically different criticism of a statement if we vary the universe of discourse (defined by the domain goals in Fickas's case). A domain goal's importance does not remain fixed but change as the analysis progresses and more knowledge about the domain is acquired. Similarly, the viewpoint models record the results of learning about the viewpoints. They need, therefore, to be updated as the investigation evolves.

5.5 Principles of the Viewpoint Resolution Approach

Chapter 2 concludes that the existing viewpoint resolution methods share the following concerns:

- definition of a viewpoint
- reasoning within a viewpoint
- reasoning between different viewpoints
- revising a viewpoint

In this thesis a **viewpoint** is defined as a source of information identified by a person who interacts with the system. There is no pre-defined criteria for selecting viewpoints, therefore allowing flexibility.

The **reasoning-within-viewpoint** problem can be stated as follows: given a piece of information:

1. how to establish its importance (relevance) to act upon it and how much efforts should be invested in an enquiry should the information turn out to be problematic
2. how to assess the credibility of the information and much to believe it
3. how to detect problems in relation to existing information from the same viewpoint and with respect to the viewpoint's model
4. how much efforts should be invested in an enquiry should the information turns out to be problematic
5. how to carry out an enquiry

This problem is addressed through the following heuristics (see previous section):

- Importance Analysis heuristics to ensure that we do not spend much time and effort on information which is not of interest and help deciding whether to pursue problems if the information is not important enough. In some cases one may take an interest in information not because the information is interesting, but because one want to find out something about the viewpoint. If the matter is important one may want to get some confirmation if there are doubts about the information.
- Information evaluation heuristics. Given that the information is important the information evaluation heuristics decides how far to believe that information using both its properties and of its viewpoint. The properties of the viewpoint (ability and trustworthiness) are taken from the viewpoint model. The viewpoint model is used to assess whether the information is compatible with the expectations of the viewpoint model. The viewpoint model is also used to modify the information, either to fill gaps in the information or to modify the information to fit the viewpoint model if a claim is made by the viewpoint which is too strong considering the known level of ability that the viewpoint possesses. Finally, the occurrence of a problem such as an inconsistency between the viewpoint model and the information may uncover more information about the viewpoint. Thus for example if an engineering configuration manager

makes a very strong claim about wordprocessing in very technical terms that will not fit our viewpoint model as we do not expect him to have expertise in that field then an enquiry may suggest that he has some qualifications in that subject. When considering the properties of the information, the information evaluation heuristics concentrate on the relative strengths of the conviction, whether the viewpoint is committed, and whether the viewpoint has vested interests if the information is acted upon.

- Single-viewpoint, conflict analysis heuristics . Once a piece of information is reconciled with the viewpoint model, conflict analysis heuristics attempt to accommodate it in the existing information, supplied by the same viewpoint. If there is a conflict then, before trying to resolve it, the importance analysis heuristics are used to decide whether it is worth pursuing the problem or the decision could be delayed until more information is available.

- Enquiry heuristics. They general are guidelines to assist finding more information about a viewpoint or establishing the roots of a given problem. For example, if there is a conflict between two viewpoints about the same issue and there is no trust, then one should look at the relation between the viewpoints and whether there are common or competitive interests, before exploring the possibility that there is a problem of abilities. Before an enquiry is launched there is quick importance analysis to justify the enquiry. For example,

The **revising-viewpoint** problem can be stated as follows: given the results of reasoning within and between viewpoints how to re-evaluate the viewpoints involved in the light of that new evidence. The viewpoint re-evaluation heuristics deal with the revising-viewpoint problem. The viewpoint re-evaluation process is interested in assessing the ability and trustworthiness of a viewpoint according to the following principles:

- If there is no previous record of the viewpoint in question then the process considers whether a classification can be produced quickly, and used as a default

- if the source is known then the process concentrates on whether there is a pattern emerging in the behaviour which should be reflected in the respective indices of its viewpoint model

- if there is hard evidence from an enquiry then it may be enough to change the viewpoint model if the viewpoint model was found to be incorrect
- if there is evidence to confirm or replace defaults then the defaults should more readily be replaced by indices labelled as originating from actual experience with the viewpoint.

The **reasoning-between-viewpoints** can be stated as follows: given a piece of information

- how to decide whether it is relevant going into conflict analysis
- how to detect problems in relation to existing information coming from different viewpoints
- how to resolve conflicts

The multiple-viewpoints conflicts analysis heuristics deal with this problem. They use the same principles as the single-viewpoints conflicts analysis heuristics.

5.6 Summary

This chapter has presented an approach that regards viewpoint resolution as a belief formation exercise of identifying viewpoints, reasoning within a viewpoint, reasoning between different viewpoints, and revising a viewpoint. The approach stresses the role of uncertainty in the information acquisition process and the crucial role that human factors and relations play in dealing with the uncertainty, should those factors were made explicit. It is based on the principle that in order to make sense of a domain one must learn about the information sources.

The principles of the new viewpoint resolution approach stem from the adaptation of the Source Control Mechanism to requirements elicitation. It has been shown that the Source Control Mechanism can be applied successfully to domains where there is a continuous flow of information from

human sources, operating in the same domain and where precision is of less importance than establishing evidence about the situation under analysis. The SCM and requirements elicitation share the principles of a court investigation where different witnesses may have conflicting or corroborating views.

The next chapter examines the use of the viewpoint resolution principles, described here, as means for the very early validation of requirements. A method called the Viewpoint Control Method is described. It is shown in the remaining chapters that by concentrating on the human factors and relations in viewpoint resolution the requirements validation method proposed by Leite is improved.

Chapter 6

Validation Through Viewpoint Control

While requirements engineering is about building a conceptual model of part of reality, requirements validation involves maximising our confidence that the resulting conceptual model 'mirrors' the stake holders' original intent. In particular, validation involves assessing the model for correctness, completeness, and internal consistency. This chapter describes a new approach that uses viewpoint resolution, described in the previous chapter, as a means for very early validation in the process of requirements elicitation. The Viewpoint Control Method, proposed here, is driven by a collection of domain-independent heuristics to build internal models of the viewpoints that record their performance in providing information, to decide whether or not to take interest in a particular piece of information, in assessing information, in resolving conflicts between different viewpoints, in enquiring to produce further information, and in re-evaluating the viewpoints. Section 6.1 introduces the concepts involved in the Viewpoint Control Method. Section 6.2 gives an overview of the method steps. Section 6.3 describes the method in details. Finally, section 6.4 gives a summary of the method.

6.1 Definitions

Our starting point towards the application of the viewpoint resolution approach to requirements validation will be to regard validation as a belief formation process. A **belief** is used to indicate the level of support that one assigns to a statement. Its operational meaning is defined by a set of endorsements qualifying it [24].

An **event** is the basic unit for validation, that is to which a belief is assigned. LOLITA processes a piece of information and translates it into a series of connected events. For example, the statement 'the system notifies the staff' is represented by LOLITA as follows *:

```
   * event: 29071 *
universal_:
  event - 7688 - rank: universal - definition_

subject_:
  system - 29069 - rank: individual - suspended_

action_:
  notify - 4639 -

object_:
  staff - 29070 - rank: individual - suspended_

time_:
  present_ - 20989 -

date:
  26 September 1993

viewpoint_:
  Roberto  - 19845 - rank: named individual

status_:
  suspended_ - 29025 -
```

The set of events with attached beliefs is called the **belief base**. The belief base, also called the

*a number attached to a node is the internal representation of that node

world model, represents the world in view of the belief formation process, as shown in Figure 6.1.

A **Viewpoint Model** is a structure that captures a record of a viewpoint. The record includes the ability, goals, trustworthiness and helpfulness of the viewpoint. A viewpoint model for a company's secretary may look like:

```
secretary:
  Ability:
     expertise: secretarial_work
     experience: high(default)
     reasoning: high(d)
  Beliefs: none(d)
  Goals: time-saving

  secretary ----> manager?
     Helpfulness: high(d)      (helpfulness of the secretary
     Trustworthiness: high(d)  towards the manager)

  manager ----> secretary?
     Helpfulness: high(d)
     Trustworthiness: high(d)  (trustworthiness of the
                                 manager towards the secretary)
```

A **universe of discourse** is the set of the existing viewpoints together with the current viewpoint models. The universe of discourse sets the context in which the information is validated (see Figure 6.1).

The results of the analyses carried out by an activity is passed on to other activities through **cases**. A case records the 'verdict' accumulated from the different analyses of a particular event. It takes a form like:

```
Case1:
   Event: event1
   Viewpoint: viewpointA

   Importance Analysis
```

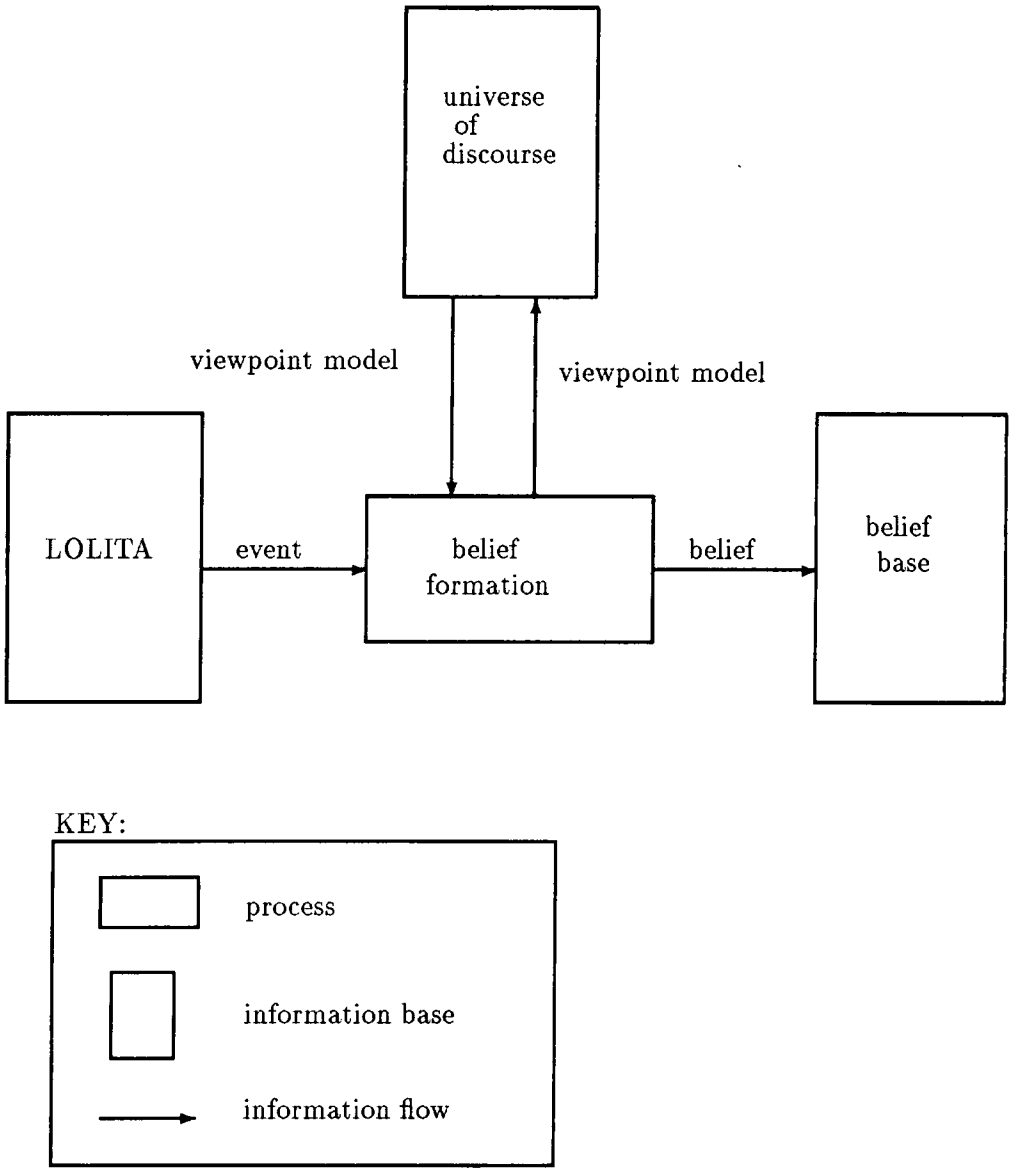

Figure 6.1: Validation as Belief Formation

```
    Viewpoint: yes
    Information: may be

Information Evaluation
    Determination: ok
    Problem of Responsibility: no
    Problem of Advantage: no
    Problem of Ability: no
    Problem of Trust: no
    Result: believe as given

Conflict Analysis:
    Event: event3
    Viewpoint: viewpointB
    Same Context: no
    Type: reinforcement
    Problems of Trust: none
    Problems of Ability: none
    Result: reinforce belief and add viewpoint

Viewpoint Re-evaluation
    Classification: clerical?  (the viewpoint is probably
    Expertise: none              of class clerical)
    Reasoning: average (default)
    Judging information: average (d)
    Experience: high (d)
    Beliefs: none (d)
    Trustworthiness: average(d)
    Helpfulness: average(d)
    Result: may be clerical classification
```

Cases act as links between the universe of discourse and the information base. Thus they allow tracing information back to their viewpoints. They also make it easier to look up the general results and problems of a previous stage of analysis to use them as a guide for further analysis and as a means to take an immediate decision if necessary. Entries may also be modified in the light of results from further analyses and enquiries.

6.2 Overview of the Viewpoint Control Method

Figure 6.2 depicts the architecture of the operational model underlying the Viewpoint Control Method. The guide coordinates the different activities of the method; it operates on a 'start-do-stop' basis. The guide is represented by a set of heuristics to assist an analyst in deciding what to do next, that is, deciding what type of analysis heuristics need to be activated at any point of the investigation. The guide needs to pass on *cases*, *events* and *viewpoint models* from one process to the next so each process has the necessary data to work on and can use the results of previous evaluations in its analysis. The control heuristics look like:

```
If there is a new case,
    then request an initial importance analysis

If the importance analysis shows that
      the information or viewpoint is interesting,
         then request information evaluation
         else store information and evaluate when required

If the information evaluation shows that
      the interest in the information is greater than
      the problems with it,
         then request conflict analysis

If there are problems and the case is important
     then try to do an enquiry by communicating with viewpoints,
      else try to find an explanation by introspection

If there are problems and the case is not important,
     then return result so far and store information
```

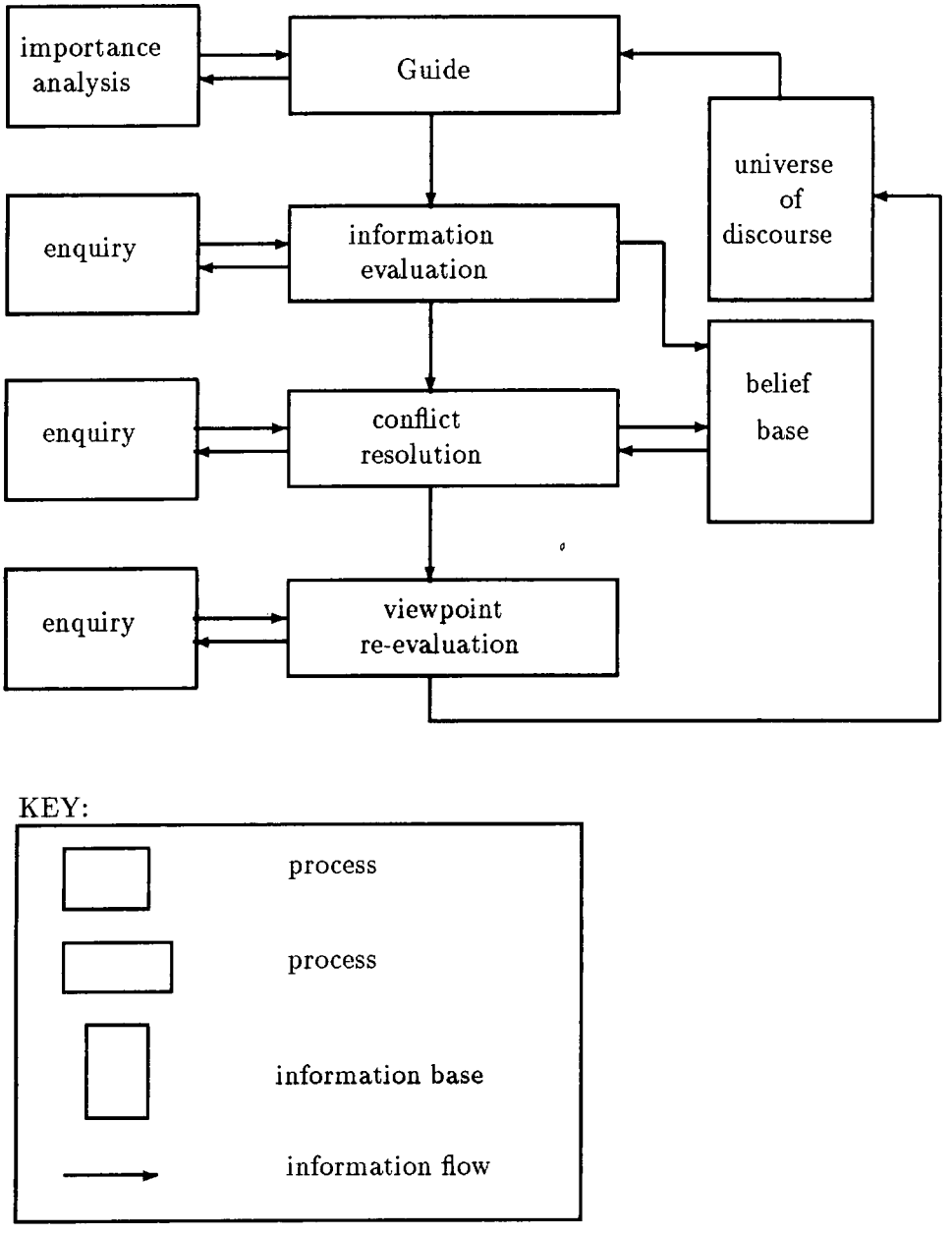

Figure 6.2: The Viewpoint Control Model

```
If after the conflict analysis there are problems involved
   then try to find out more
   else return the result

If the case has been completely analysed,
   then request a viewpoint re-evaluation

If the viewpoint re-evaluation is having problems
   and the viewpoint is of above average importance,
      then try to enquire to find an explanation
```

Importance analysis will try to see whether the case is important for the investigation. The check should be fairly shallow to see whether there is a reason to go on. As a result, the information will either proceed to the next stage or be expelled immediately. The importance will come from the analyst's motivation towards the viewpoint or because there is an interest in the information.

The information evaluation checks if there is a problem with the information, and if so considers whether the case is worth pursuing further. If the problem is serious, in the sense that it needs investing serious efforts to solve, then the case needs to be sufficiently important so the problem can be investigated.

Conflict analysis evaluates the information in relation to the existing information, either from the same viewpoint or from a different one. Again, importance analysis must establish whether it is worth going into conflict analysis and the subsequent resolution of conflicts and enquiries.

The enquiry can occur at any stage of the investigation where more insight, about the information under analysis or about the viewpoint, is needed. The enquiry can either exploit the existing information or ask the viewpoints for more information.

Once the analysis has finished the results are passed to the viewpoint re-evaluation process, which has to consider the viewpoint models in the light of new information and to decide whether it is necessary to adjust them.

6.3 The Viewpoint Control Activities

The Viewpoint Control Method comprises the following activities; Figure 6.3 shows a SADT model of the method (the figure shows the life histories of an event and a viewpoint model):

- Universe of Discourse Initialisation
- Information Validation
  - Importance Analysis
  - Information Evaluation
- Communication
  - Conflict analysis [†]
  - Enquiry
  - Universe of Discourse Update

Figure 6.3 is decomposed in Figure 6.4 and Figure 6.5. The Viewpoint Control activities are driven by a collection of domain-independent heuristics to decide whether or not to take interest in a particular piece of information, in assessing information, in resolving conflicts between pieces of information, in enquiring to produce further information and also in reevaluating the corresponding viewpoint models.

[†]Conflict analysis and conflict resolution are used interchangeably

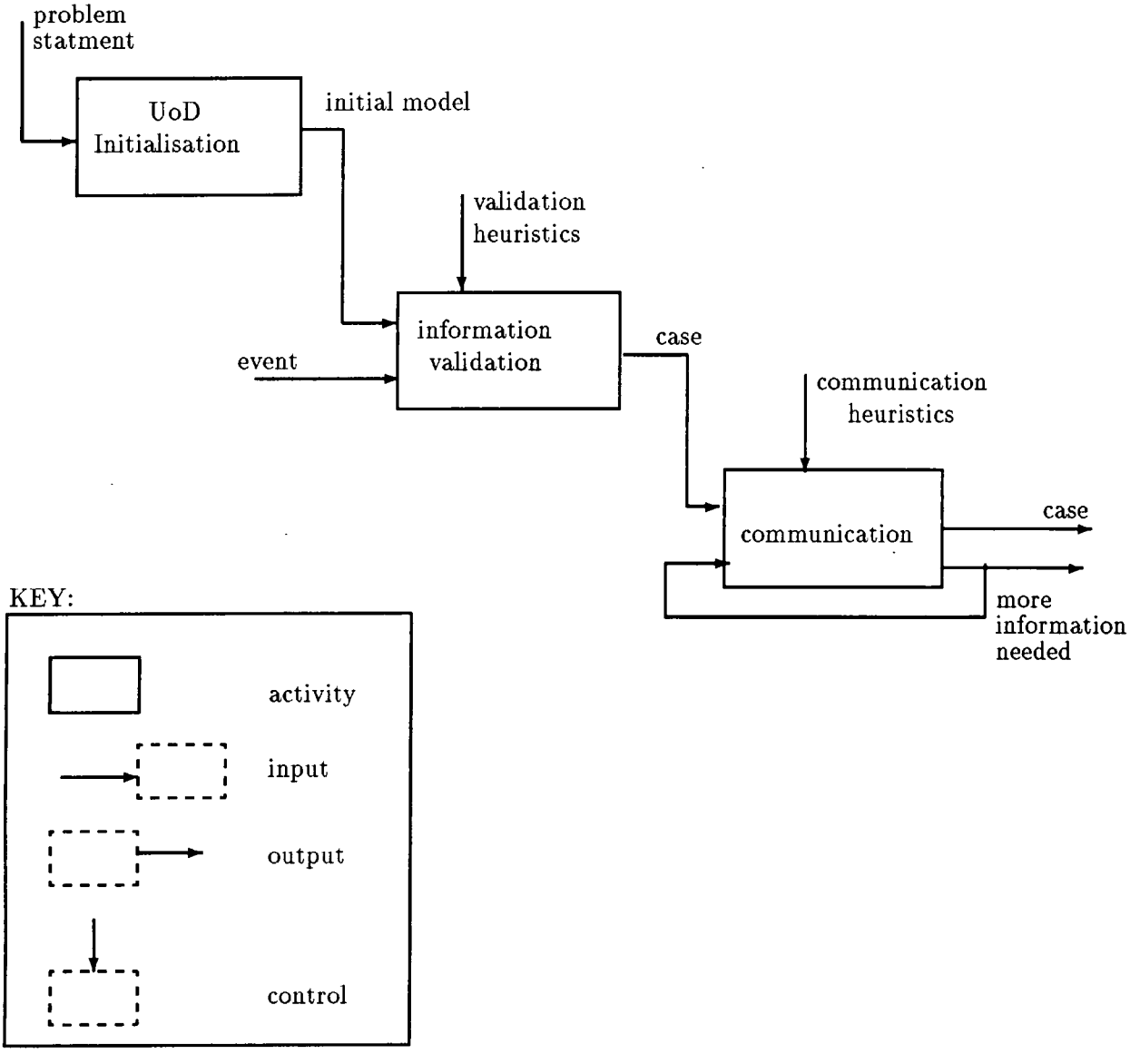

Figure 6.3: The Viewpoint Control Method

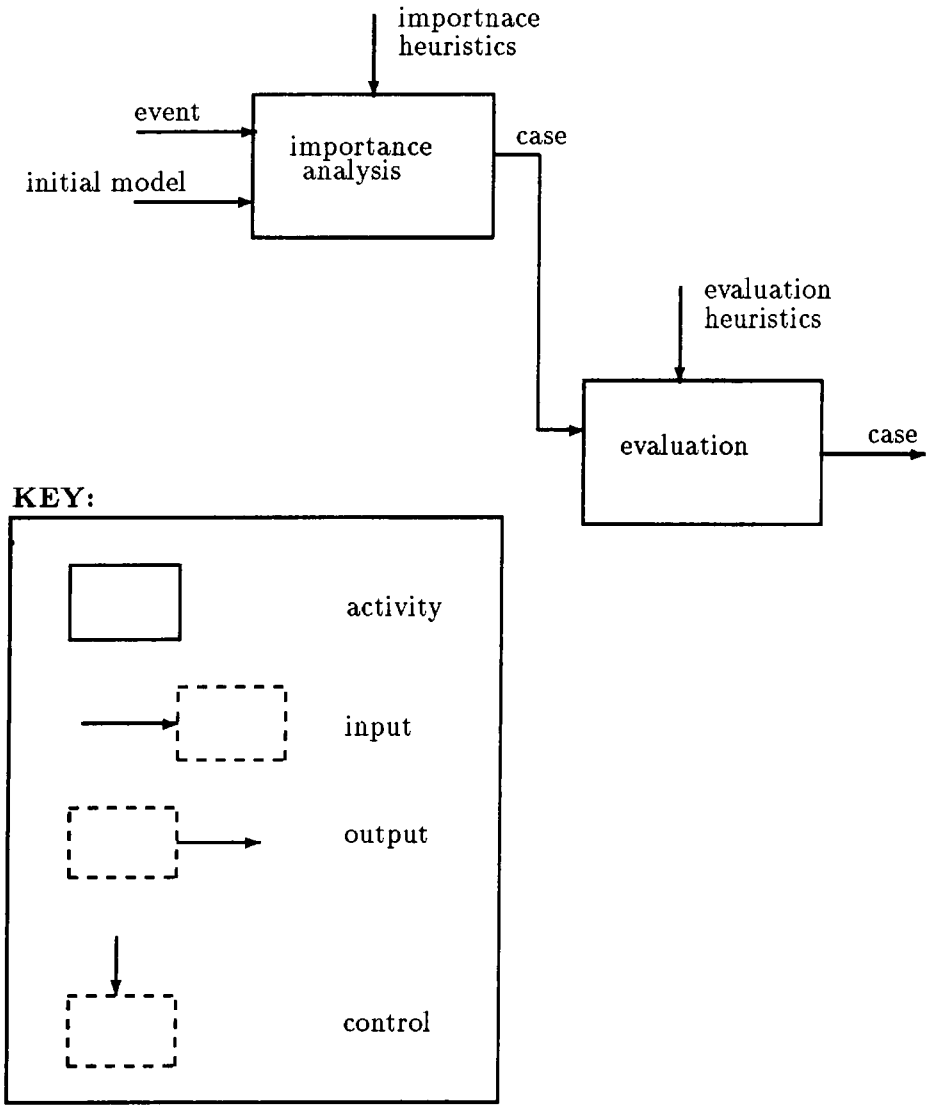

Figure 6.4: Information validation

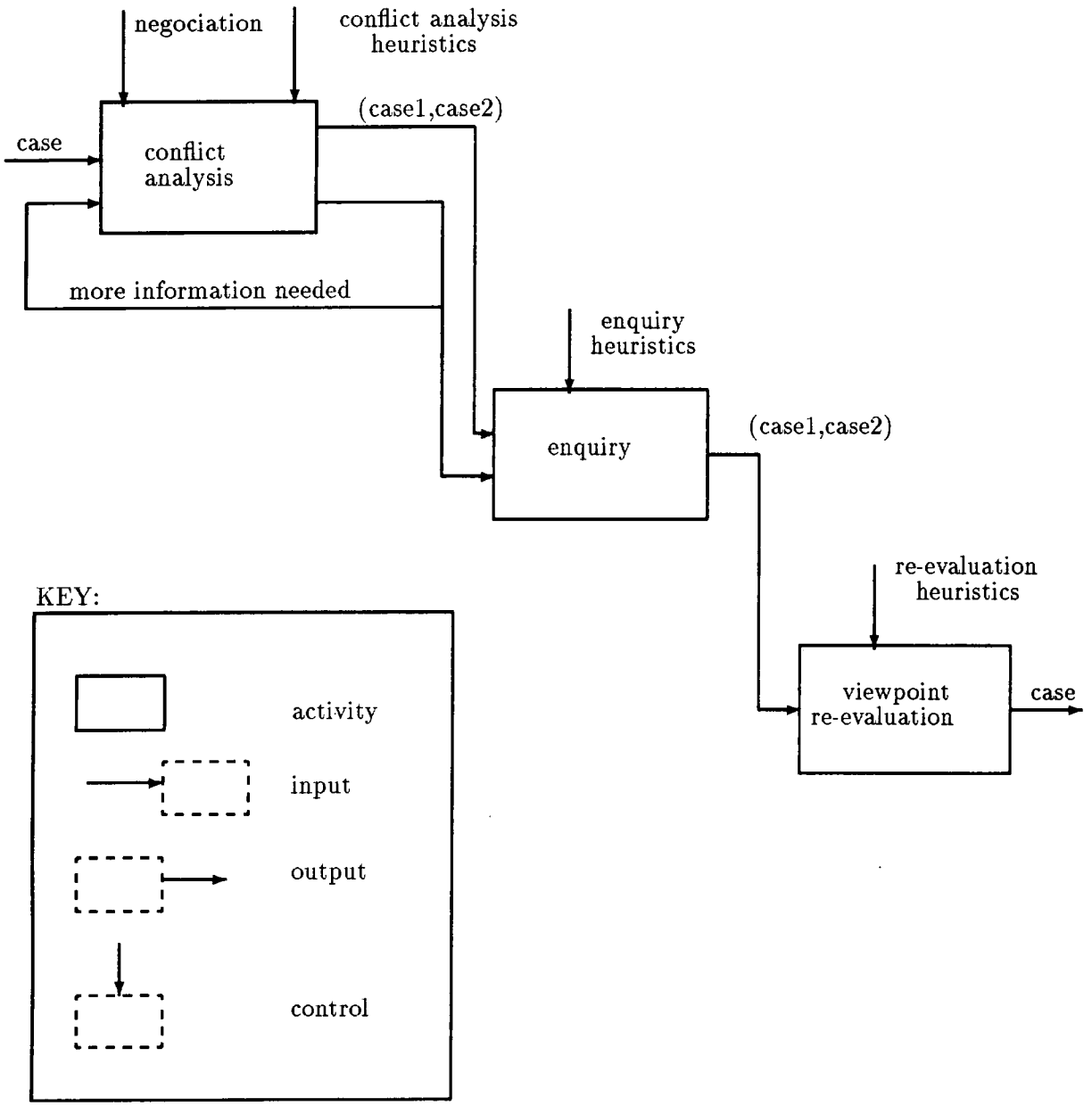

Figure 6.5: Communication

These heuristics operate on a number of parameters (the entries in a *case*) some of which are immediately available from the events under analysis and others are implicit and need to be extracted using the available information. These parameters have been explained in the previous chapter. For example, to identify the degree of *commitment* one may use the following heuristic:

```
if Commitment not indicated and trustworthiness is average,
   then
      if helpfulness is low and information volunteered
          then Commitment suspect
      elseif information asked for
         then Commitment expected
```

6.3.1 Universe of Discourse Initialisation

Establishing the universe of discourse requires selecting the relevant viewpoints that will take part in viewpoint resolution and establishing the relationships between them. The relationships are represented in a viewpoint hierarchy. A viewpoint hierarchy may represent membership relationships, e.g. a person belongs to a department, reporting relationships, e.g. who reports to whom? where do they get information from? etc., and ownership relationships, e.g. a person supervises another person or a team. Due to the explorative nature of requirements elicitation, it is difficult to establish what the relevant viewpoints and their properties are, before the acquisition process begins. As a starting point, an initial set of viewpoints is defined. Initial viewpoint models are then constructed using a default and a classification mechanism by which default values of viewpoint models can be produced and used in the absence of concrete evidence. Viewpoints which can be associated with a particular class are assumed to have the typical properties of that class. These class defaults are then used until further evidence either confirms or rejects them. For example, a *doctor* and a *nurse* may be selected as initial viewpoints of a patient monitoring system that is responsible for notifying the staff of an abnormality in the conditions of an intensive care patient.

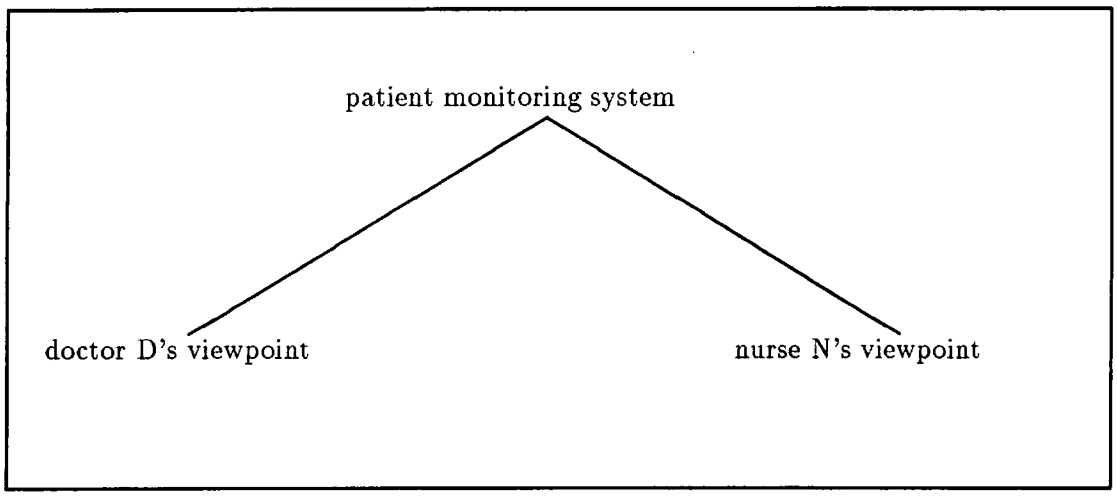

Figure 6.6: A two-level viewpoint hierarchy

Later, another member of staff is added if necessary. The notation in Figure 6.6 means that the views on one level of the diagram are subviews of the parent view on the higher level.

```
Doctor D:
  Ability:
     experise: General_Medicine(d)
     experience: 3_year_practice
     reasoning: high(d)
  Beliefs: none(d)
  Goals: ?

  doctor ----> nurse?
     Helpfulness: high(d)
     Trustworthiness: high(d)

  nurse ----> doctor?
     Helpfulness: high(d)
     Trustworthiness: high(d)
```

It must be noted that the viewpoint model contains only information which is relevant for the purpose of assessing information. Thus, if a viewpoint has a strong belief which does not impair its subjectivity, then that will not be represented in the viewpoint model.

6.3.2 Importance Analysis

The role of importance analysis is to decide how far the analyses should go. There are three types of importance analysis: pre-processing importance analysis, pre-conflict importance analysis, and pre-enquiry importance analysis. Given an event and the corresponding viewpoint model the pre-processing importance analysis recommends:

1. proceed to information evaluation
2. accept the information as given, or
3. reject the information as irrelevant.

Pre-conflict importance analysis recommends one of the following:

1. proceed to conflicts analysis, that is to analyse the event in the context of the existing information.
2. launch an enquiry to further analyse the current event in isolation
3. stop analysing the event any further.

Pre-enquiry (post-conflict) importance analysis recommends one of the following:

1. enquire not
2. enquire about the viewpoint

3. enquire about the information

4. re-evaluate the viewpoint and investigate

6.3.3 Information Evaluation

Information evaluation assesses a piece of information both in isolation and against the universe of discourse. The ultimate objective of the information evaluation process is to reach a decision on how much credibility can be attributed to a piece of information by considering its external features and the features of its source (viewpoint).

The information is analysed for consistency, correctness, and incompleteness. A piece of information is:

- incorrect if it is attributed a very low or a nil belief.
- inconsistent if it does not live up to the expectations of the viewpoint model or if it causes conflicts with related information
- incomplete if there is evidence of the need for more information that requires an enquiry to reach a decision about it.

The distinctive feature of this approach to validation is its exploitation of the correlation between the problems of inconsistency, incompleteness, and incorrectness to form an opinion about a piece of information, thus making the maximum use of the information available. For example, an inconsistency may provoke an enquiry to find an explanation, and the enquiry may reveal evidence that may lead to the modification of the viewpoint model which in turn affects the decision on the degree of the information reliability.

Information evaluation considers the following attributes:

- the relative strength of the argument
- the degree of the viewpoint's commitment
- the degree of the viewpoint's advantage
- the viewpoint's trustworthiness
- the viewpoint's ability - expertise, experience, etc.

The final outcome of information evaluation is one of the following recommendations:

1. accept the event as given,
2. modify belief as a function of the viewpoint's ability,
3. reject the event.

For example, given the the ability and the trustworthiness of a viewpoint the information evaluation may use the following heuristics:

```
if the helpfulness expected from the viewpoint model is low,
the trustworthiness of the viewpoint is low, and the actual
helpfulness of the viewpoint is high (i.e. the information
was not solicited, but volunteered), then the viewpoint's
advantage can be expected to be high (i.e. we can suspect a hiden
advantage or vested interest) and we can conclude that the certainty
of the information is low.
```

6.3.4 Enquiry

An enquiry is required if more information is needed. Information may be required to find out more about a particular viewpoint or about the information under analysis. There are two types of

enquiry. The post-information evaluation enquiry which is prompted by problems with the current event when analysed in isolation. The Post-Conflict Resolution enquiry prompted by the results of analysing the current event in relation to the existing information, either from the same viewpoint or from a different viewpoint. The objective of an enquiry is to re-evaluate information via different venues. An enquiry can recommends the following:

1. accept the event as given,
2. modify belief as a function of the viewpoint's ability,
3. reduce belief to below the action point, or
4. reject the event.

6.3.5 Universe of Discourse Update

Once a case has been completely analysed the viewpoint re-evaluation process takes over to revise the corresponding viewpoint model in the light of any new evidence about the viewpoint characteristics. Figure 6.7 shows that requirements validation is the composition of two processes: information evaluation and information viewpoint evaluation. The information evaluation process feeds details about the information to the viewpoint model. In return, the viewpoint re-evaluation process provides its evaluation about the viewpoints by returning revised viewpoint models. Revising a viewpoint model is to modify the information it records, namely:

1. Ability related indices:
   (a) the viewpoint's expertise in different areas
   (b) the viewpoint's reasoning capabilities
   (c) the viewpoint's competence in judging information
   (d) the viewpoint's capabilities in handling its own experience

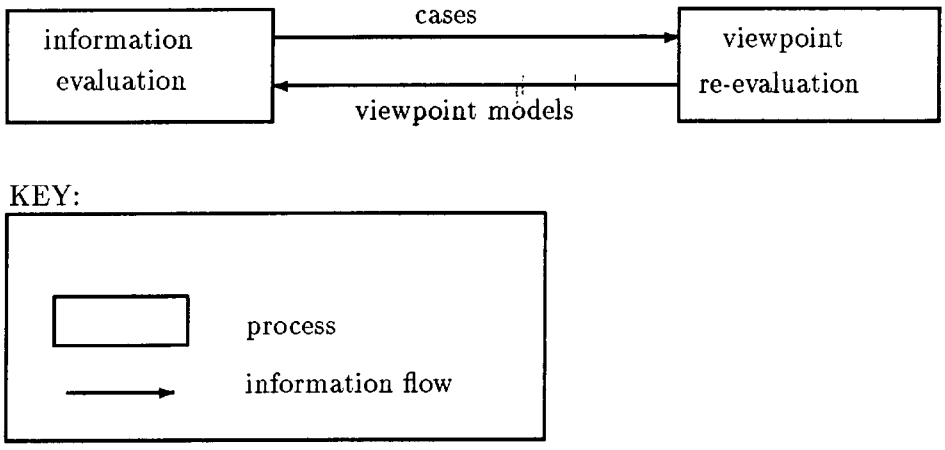

Figure 6.7: Information Evaluation vs. Viewpoint Evaluation

2. Trust related indices:

   (a) The viewpoint's fundamental beliefs

   (b) The viewpoint's goals

   (c) The viewpoint's special relationships

The re-evaluation process produces one of the following recommendations:

1. adjust index
2. split index
3. replace default index with index based on evidence
4. record evidence and stop
5. introduce a new index for the class
6. investigate

6.3.6 Conflict Resolution

If we add a new event p to the existing belief base, then a problem has arisen because of some kind of conflict between p and some other event q. If p contradicts q then there is no non-arbitrary way of choosing between them unless the 'supports' of p and q are known. The support of an event is estimated from the values of the its external attributes and from the record of the corresponding viewpoint, using some kind of attributes utility analysis. There are two cases to be considered:

- the conflicting events originate from the same viewpoint (single-viewpoint conflicts)
- the conflicting events come from different viewpoints (multiple-viewpoint conflicts)

There are four types of conflict considered here:

- contradiction - two events at odd with each other.
- reiteration - the two events are roughly identical (redundancy)
- reinforcement - the new event strengthens the old
- weakening - the new event weakens the old

Contradictions are considered to be the most relevant type of conflict. Viewpoints are not judged just on the occurrence of contradictions. Contradictions are used as a signal to trigger a more complex evaluation. The outcome of the conflict analysis process is one of the following:

- accept information as given
- modify belief as a function of the viewpoint's ability
- reject information
- expel both pieces of information

- adjust belief according to the relative strength of the viewpoints
- merge both pieces of information
- set belief at even level, i.e. low/low or medium/medium
- suspend and investigate

For example, in the case of a single-viewpoint contradiction we may have the following:

```
if there is a contradiction and both pieces of information are
of high claimed certainty
and there is a problem of trust
and there is no problem of ability
then reject both pieces of information
```

In case of a multi-viewpoint contradiction we may have the following case:

```
   if there is a case of contradiction
   and the old information is less solid than the new
   and there is no problem of trust
then suspend and investigate
```

This resolution method can be seen as a combination of judicial and extra-judicial resolution methods [45]. A judicial resolution method covers situations where a third party is called upon to take a decision, taking into account the cases presented by each viewpoint. An extra-judicial resolution method covers situations where a decision is determined by factors other than the cases presented (e.g. by the relative status of the conflicting viewpoints). However, if a deadlock is reached, there is the need to start a negotiation process in order to reach a new solution. Negotiation is a complex, iterative process of generation followed by evaluation [96]. The techniques of importance analysis,

information evaluation, enquiry, and viewpoint re-evaluation can be part of such an iterative process. Also, the information recorded in the viewpoint models and cases can be part of an 'agenda' for the negotiation. Using the fuzzy-logic based formalism [50] implemented to support interactively the parties in achieving a common solution, we can illustrate how the Viewpoint Control techniques can play a significant part in a negotiation process, especially in an uncertain environment. The conflict resolution approach can be outlined in the following steps:

1. On the basis of ability and trustworthiness of the viewpoints assign a weight to each viewpoint. These weights can be changed following re-evaluation of the viewpoints.

2. Evaluate the advantages and disadvantages of each option on the basis of the beliefs assigned to them by the information evaluation process and some domain-dependent knowledge.

3. Evaluate the consensus degree and shifting of positions:

   (a) build a matrix representing the judgement of each group on each option, expressed in fuzzy linguistic terms, i.e. each element of the matrix is the value of a linguistic variable the range of which is pre-defined. For example: V = (very low, low, medium, high, very high). The viewpoints are made aware of the weights assigned to the options (stage 2).

   (b) Measure the distance of each viewpoint from the general agreement on the value of each option, taking into account the weights placed on the viewpoints. The options may then be chosen for further discussion, and viewpoints may be asked to shift their positions.

   (c) Rank the options, and repeat the process until an acceptable solution is found or a deadlock is reached.

4. Start a negotiation process if a deadlock is reached.

Note that the importance analysis and enquiry processes can be called at any stage if needed. This is illustrated in the next chapter by case studies.

6.4 Summary

A method for the very early validation of requirements has been developed. The method provides techniques to decide the relevance/importance of a piece of information, to assess it for correctness, to detect inconsistencies, to resolve conflicts between different viewpoints, to enquire for producing further information, and to re-evaluate viewpoints in the light of the accumulated evidence with regard to the performance of the viewpoints in providing information. These processes are coordinated by a guide operating on a 'start-do-stop' basis. The Viewpoint Control has the main ingredients of a validation method. These ingredients have been described in Chapter 1 as the ability to:

- detect wrong information, inconsistencies, and missing information with respect to the universe of discourse as early as possible,
- allow for traceability between the information and the universe of discourse,
- encourage the users' involvement in the process, and to
- support the negotiation process for resolving the problems with the requirements.

Although the Viewpoint Control concentrates on the fact-validation part of the requirements elicitation process it supports the fact-finding subprocess through its importance and enquiry techniques and supports the communication subprocess through its support for the negotiation process (see Figure 1.1).

The Leite method lacks many of the above ingredients. Firstly, the method supports neither fact-finding nor communication. The elicitation subprocesses (fact-finding, fact-validation, communication) are naturally tied with each other that it becomes difficult to separate them. The Leite method is inadequate to cope profitably with the iterative and feedback nature of these processes. Secondly, within fact-validation itself the Leite method does not deal with the correctness problem, e.g. two statements may be consistent but wrong with respect to the universe of discourse.

The development of the Viewpoint Control Method has concentrated on the causes of the inadequacies in Leite's method in order to improve on it.

These causes have been identified in Chapter 3 in which the Leite method is evaluated. The causes are:

- The method depends on the quality of the viewpoints selected to take part in viewpoint resolution. Viewpoints in the Viewpoint Control Method are themselves evaluated and re-evaluated as the investigation progresses.
- The method is restricted to two viewpoints. This is not an issue for Viewpoint Control since statements from different viewpoints are integrated as the requirements evolve.
- The context (i.e. universe of discourse) in which facts are validated is not defined. As a result there are no links between universe of discourse and information. The Viewpoint Control Method uses a domain-independent universe of discourse defined by the relevant viewpoints and by the viewpoint models. Furthermore, the use of cases relates the viewpoints to the information they contributed.
- The method is unable to supply the negotiation and conflict resolution process with the 'roots' of conflicts. For the Viewpoint Control Method each statement is associated with a case and a viewpoint model; the case records the results of the 'verdict' as to its quality and the viewpoint model captures the record of its source.

Chapter 7

Application of the Method

In this chapter the concepts and ideas developed in Chapters 5 and 6 are demonstrated with an example and two case studies. The example is constructed to illustrate the kind of problems this work is addressing and to show how the proposed approach alleviates the problem that the Leite method fails to cope with. The cases studies were designed to illustrate different aspects of the Viewpoint Control Method and to determine its feasibility and its practical utility as a problem investigation and information validation technique. The studies collectively address the issues raised in the criteria for success in Chapter 1.

7.1 The Case Studies

The key criteria for selecting the case studies are the different kinds of uncertainty that are involved, namely:

- a changing environment/context,

- different people (different skills, goals, commitments),
- many sources of information,
- various constraints, such as time, money, etc.

The first case study illustrates the investigative nature of the method and the effectiveness of its heuristics in pointing out problems and in detecting pointers to missing information.

The second case study concentrates on the communication part of the method: to support group decision under uncertainty. In particular, the study shows how the information produced by the method can help the negotiation of compromises between competing views.

The example problem is one of the examples used in Chapter 3 to evaluate the Leite approach.

Designing case studies that illustrate both the inadequacies of the Leite method and the improvements the Viewpoint Control has made is problematic for the following reasons:

- the two methods have different scopes. One of the objectives of the Viewpoint Control Method is to have the ability to detect problems earlier than the Leite method allows and the ability to support the negotiation process. This means that the Viewpoint Control Method had to include the process that goes before the final views/perspectives are ready to be compared for consistency. It also means that the Viewpoint Control Method needs to identify viewpoints and define the universe of discourse. This process is taken for granted by Leite;
- the Viewpoint Control Method concentrates on the external features of the information (who said it, how it is said, etc.) while the Leite method concentrates on its structural features;
- the Leite method operates under numerous assumptions, some of which are unrealistic.

The example and the case studies were carefully selected to achieve both objectives. To complete the demonstration, the Viewpoint Control Method is compared to Leite's using the same comparison

scheme that Leite used to evaluate his method as a validation technique. This is shown in the next chapter.

7.2 An Example Problem and its Solution

The example concerns a system for monitoring a high security intensive care unit. The monitoring system should poll sensors at regular intervals. When sensors detect undesirable events the system notifies the emergency services of exceptional conditions. The following are two possible interpretations of the problem, from two different viewpoints. The viewpoints are represented by the analysts A and B:

ViewpointA

The system polls a sensor every 60 seconds. If the sensor detects an abnormality the system notifies the emergency services of exceptional health conditions.

ViewpointB

The system polls a sensor every 60 seconds. When the sensor detects an abnormality the system notifies the emergency services of exceptional security conditions.

Suppose that the analysts use the methods CORE and Remora, respectively, to establish facts about the problem. Using Leite's method, possible codifications of ViewpointA and ViewpointB into VWPL (VieWPoint Language) are:

```
            ViewA
3 (notify =alarm-buzzer-sounding) (exceptional-condition =patient)
(patient-monitoring-system =monitoring-data)
(sensor =location =type =60-seconds)
------>
```

```
$delete from wm (notify =alarm-buzzer-sounding)
$add to wm (alarm =alert-alarm-state =audi-visual-alarm)
           (emergency-services =health}

Hierarchies
(is-a (emergency-services doctor)
(parts-of (monitoring-data blood-pressure temperature))


             ViewB
4 (notify =alarm-sounding) (exceptional-condition =alarm)
(alarm-monitoring-system =monitoring-data)
(sensor =location =type =60-seconds)
------>
$add to wm (alarm =audi-visual-alarm) (emergency-services =security)
$delete from wm (notify =alarm-sounding)

Hierarchies

(is-a (1 (alarm-state alert-alarm-state safe-alarm-state)
         (emergency services police firemen)
(parts-of (monitoring-data lights)
```

Consider, for example, the statement 'the system polls a sensor every 60 seconds'. It appears in both viewpointA and ViewpointB. The statement refers to a critical non-functional requirement for a real-time system and if faulty it may lead to disastrous consequences. However, there is no information available to help in deciding about its correctness.

As shown below, the Viewpoint Control Method copes with such situations. Using the method the above statement will be subjected to a 'scrutiny'.

The following is an extract from LOLITA's semantic interpretation of ViewpointA. The node viewpoint in * event 29073 * is instantiated to 'roberto' by default.

```
**********************************

    * event: 29073 *
 universal_:
   event - 7688 - rank: universal - definition_

 subject_:
   system - 29069 - rank: individual - suspended_

 action_:
   poll - 27643 -

 object_:
   sensor - 29070 - rank: individual - suspended_

 time_:
   present_ - 20989 -

 date:
   26 September 1993

 viewpoint_:
   roberto - 19845 - rank: named individual

 status_:
   suspended_ - 29025 -

**********************************

    * event: 29075 *
 universal_:
   event - 7688 - rank: universal - definition_
 cause_of:
   event - 29082 - rank: universal - hypothesis_

 subject_:
   sensor - 29070 - rank: individual - suspended_
```

```
 action_:
   detect - 15148 -

 object_:
   abnormality - 29074 - rank: individual - suspended_

 time_:
   present_ - 20989 -

 date:
   26 September 1993

 viewpoint_:
   roberto - 19845 - rank: named individual

 status_:
   hypothesis_ - 21034 -

**********************************

    * event: 29082 *
 generalisation_:
   event - 7688 - rank: universal - definition_
 cause_:
   event - 29075 - rank: individual - hypothesis_

 subject_:
   system - 29069 - rank: individual - suspended_

 action_:
   notify - 4639 -

 object_:
   service - 29078 - rank: universal

 time_:
   present_ - 20989 -

 date:
   26 September 1993

 viewpoint_:
   roberto - 19845 - rank: named individual

 status_:
   hypothesis_ - 21034 -

**********************************
```

The method does three things:

1. finds out about the background of the statement, e.g. who said it (their experience, expertise, trustworthiness, etc), does the source accept responsibility for the consequences of such a statement, is there an advantage for the source, etc.
2. uses that background to judge the statement,
3. takes action. The result of the analysis is used not only to reject the statement if it turned out to be not credible but also to revise its source's record (i.e. update the viewpoint model).

Suppose that the information is first supplied by the viewpointA represented by the doctor D. The analysis can either proceed by considering the characteristics of the information, the characteristics of its source or both. There exists, in the universe of discourse, information about the source. The information is recorded as a the viewpoint model:

```
Doctor D:
  Ability:
     expertise: General_Medicine
     experience: 3_year_practice
     reasoning: high(d)
     judging sources: high
  Beliefs: none
  Goals: ?

  doctor D ----> nurse N?
     Helpfulness: high(d)
     Trustworthiness: high(d)

  nurse N ----> doctor D?
```

```
Helpfulness: high(d)
Trustworthiness: high(d)
```

The model records general abilities of doctor D and the doctor's relations to the viewpoint 'nurse N'. The viewpoint model can also contain the relations of the viewpoint 'doctor D' to the analyst carrying the investigation.

We can assume that the analyst choses to use the external characteristics of the statement as well as information from the viewpoint's record. In this case the analyst needs to determine the relative strength of the viewpoint's conviction, whether doctor D accepts responsibility for the consequential risks of the statement, and whether the viewpoint could derive an advantage out of the analyst's acting on the statement.

The Viewpoint Control Method uses heuristics to determine the different parameters and to use those parameters to make a decision. For example:

```
if responsibility is denied or there is an advantage
      then there is a problem of trust

If the conviction in the information is high
    and there is a problem of trust
    and there is no problem of ability
      then assign a low belief (not for use)
```

Later, the same statement would be supplied by the second viewpoint (e.g. nurse N). In a similar manner the model of the viewpoint 'nurse N' would be consulted and used to evaluate the statement from the nurse's viewpoint. Once evaluated in isolation the statement is judged with respect to the existing information. In this case we have a conflict of type reiteration. The method also provides guidelines on how to handle such situations. The final outcome is an evaluation of the statement from two different viewpoints and the re-evaluation of the viewpoints involved.

In summary, the problem can be solved with little difficulty by applying the Viewpoint Control Method, as it records who said what, and what evidence is given to endorse what was said.

7.3 Case Study 1

7.3.1 Has the Manager a Business Case?

> An engineering configuration manager decides to purchase a word processor for his secretary claiming that it will improve the quality of the control construction. From the secretary's point of view the word processor will save time. The item's cost is estimated at 7,000 pounds - a year's salary for a secretary. The problem for the financial department is to establish whether the manager has a business case or merely wants a new toy and prestige for his secretary. Time and resources constraints are taken into account.

7.3.2 Initial Universe of Discourse

From the problem statement there are four important viewpoints: the manager, the secretary, the financial department, and the analyst. Using the available information, the analyst first creates initial viewpoint models for the manager and the secretary:

```
manager
   Ability:
      expertise: engineering configuration management
      experience: high
      reasoning : high
      goals: improve the  quality of the control construction

  manager ----> analyst
      helpfulness : average(d)
      trustworthiness : high(d)
```

```
  analyst ----> manager
      helpfulness : high(d)
      trustworthiness : average(d)

secretary:
  Ability:
     expertise: administration
     experience: high
     reasoning: high(d)
     goals: save time?

 secretary ----> analyst
     helpfulness: average
     trustworthiness: suspicious

analyst ----> secretary
     helpfulness: high
     trustworthiness: impartial
```

7.3.3 Importance Analysis

Given the high cost of the item ordered and the position of the manager the analyst has to justify both to the financial department and to the manager the acceptance or rejection of the order. This means any piece of information related to the business situation will be considered as important.

7.3.4 Information Evaluation

The analyst has to evaluate the manager's statement that a word processor improves the quality of his department services. This can be represented by LOLITA as the event:

```
   * event1 *
universal_:
  event - 7688 - rank: universal - definition_
subject_:
  processor - 29069 - rank: individual - suspended_
action_:
```

```
  improve - 16916 -
object_:
  quality - 29071 - rank: individual - suspended_
time_:
  present_ - 20989 -
date:
  26 September 1993
viewpoint_:
  manager - 19845 - rank: named individual
status_:
  suspended_ - 29025 -
```

The analyst finds out that although the manager has good managerial expertise, including budget management, he lacks the appreciation of some of the new technology. The analyst would assign a low belief to the information according the following heuristics:

```
if the information requires expertise
   and the conviction is high
   and the ability in that subject is low          (1)
     then problem of ability ----> yes

if the conviction in the information is high
   and there is a problem of ability               (2)
      then belief ----> low
```

Suppose that the analyst decides to consider the problem from the secretary's viewpoint. The analyst has to assess the fact that the word processor will save the secretary's time by first consulting the secretary's model.

Similarly, the analyst concludes that the secretary is behaving irrationally (i.e. reasoning=low) as he is trying to assume technical expertise in the word processing field.

The analyst may choose, instead, to evaluate the information from the secretary against the existing information. In this case the secretary's statement that the word processor saves time is a reiteration of the manager's statement that the word processor improves the quality of the department's services, assuming that saving time is an improvement.

Assuming that the convictions for both statements are relatively high, the analyst gets to the same conclusion (attach low belief to the secretary's statement) using the following heuristic and the results of heuristics (1) and (2):

```
if the old conviction is equal to the new
   and the conviction in the new information is high     (3)
   and there is no problem of trust
   and there is a problem of ability,
       then check out ability problem
```

Suppose that the analyst decides to gain a first hand experience of how much typing the secretary does in a typical working day. He notices that the secretary spends only one hour and a half typing and spends the rest of the day doing other administrative work (note that `event3` below is an example and not an actual output of LOLITA):

```
event3:
   subject: secretary
   action: type
   status: real
   time: present
   viewpoint: analyst
   certainty: high (real)
```

Let's assume that the information from the analyst is credible and does not need further analysis.

At this stage the analyst detects a contradiction between the analyst's statement (`event3`) and and the secretary's statement. The analyst could decide to suspend `event2` and investigate because `event3` is stronger (based on first hand experience) and because of the following heuristic:

```
if there is a contradiction                              (4)
   and the new information is stronger than the old,
  then suspend and investigate
```

One may argue that `event3` *weakens* `event2`. Thus, assuming that there are advantages on the part of the secretary in trying to convince the analyst, the result is the need for an enquiry according the following heuristics:

```
if a piece of information weakens another then the analyst has to check
the relative weight of viewpoints and whether there are interests and
advantages involved if they are in the same context:
                                                                    (5)
if there are interests and advantages then the analyst will be
inclined to start an enquiry unless that is not possible. In
the enquiry process independent viewpoints should take precedence.
```

7.3.5 Enquiry

Suppose that the analyst decides to establish a profile of the secretaries around the whole company. The analyst discovers the existence of a *design services pool* (DSP). The analyst finds out from its supervisor that the DSP does the typing (seven hours a day) for all the departments, including the engineering configuration department. The analyst is interested in establishing whether the introduction of a word processor to the DSP would improve its productivity.

The analyst can establish this simply by using a mathematical model from operational research (e.g. queuing theory) as a way of checking if the introduction of a word processor cuts down the waiting list for the DSP. Suppose that the analyst concludes that a word processor does cut down the waiting list for the DSP.

The information from the DSP manager weakens the department manager's claim. This leads to further reduction in the credibility of that claim:

```
if the new information is stronger than the old,
     and there is no problem of trust or ability (of the DSP manager),
       then reduce belief (dept. manager) and add the new viewpoint
```

7.3.6 Conflict Resolution

The obvious decision is to reject the department manager's claim because it is incorrect (very low belief assigned). However, this is not the case in practice, according to the general heuristics:

```
The department manager can not be ignored.

A solution that leaves any group totally
unsatisfied can not be accepted.

Elements from each view have to be contained
in the proposed solution.

A qualified majority is required.
```

Thus, there is a need for find a compromise through negotiation between the manager, the financial department and the DSP as well as the analyst. The analyst may propose that the word processor should be purchased for the DSP with one redundancy from that group.

The Viewpoint Control Method does not deal with the negotiation process but the information recorded in the viewpoint models and the cases is meant to be used as part of an agenda for the negotiation. This is illustrated by the second case study.

7.3.7 Universe of Discourse Update

In this case the DSP supervisor viewpoint is added to the universe of discourse together with an initial viewpoint model. Then the manager viewpoint model needs to be updated in the light of

the new evidence from the enquiry. A possible parameter to be updated in the manager's model is the 'goals' parameter. Initially, the manager's goal was set to: 'improve the quality of the control construction'. Following the investigation, the goal could be changed to 'gain prestige'. Other defaults values in the viewpoint models could also be confirmed or modified.

The results of the analyses can be recorded in the following case:

```
case10:
        Event: event1
        Viewpoint: manager

         Importance Analysis
            Viewpoint: yes
            Information: yes

         Information Evaluation
            Determination: ok
            Problem of Commitment: no
            Problem of Advantage: yes
            Problem of Ability: yes
            Problem of Trust: no
            Result: low belief

         Conflict Analysis:
            Event: event2
            Viewpoint: secretary
            Same Context: yes
            Type: reiteration
            Problems of Trust: none
            Problems of Ability: yes
            Result: keep belief at present level

         Viewpoint Re-evaluation
            Goals:??
            Expertise: low
            Reasoning: high
            Judging information: high
            Experience: high
            Trustworthiness: high
            Helpfulness: average
            Result: gain prestige
```

7.4 Case study 2

7.4.1 Route Generation and Selection

The term 'route' is used in manufacturing industry to mean the design and production phases [84]. The phases are strongly linked since the characteristics of a design determine the manufacturing processes needed and features of the production cycle act as constraints on the acceptable designs. The traditional approach to this problem is sequential and is inadequate to cope profitably with the feedback nature of these processes, which are naturally iterative and interactive.

Thus, to reduce lead time and costs and to improve communication it is necessary to adopt a Simultaneous Engineering approach that allows designers and production engineers to work in parallel and synchronously.

7.4.2 Universe of Discourse Initialisation

The initial list of viewpoints is made up of the Simultaneous Engineering Team (SET),

```
SET = (V1,V2, ... , Vk)
```

For example

```
V1 = group of production engineers
V2 = designers
V3 = accountants
V4 = personnel manager
V5 = marketing manager
etc.
```

Given a common task (e.g. marketing) of each group it is possible to construct an aggregate viewpoint model (ability, trustworthiness, goals, etc) for each group using information that can be obtained from official documents, interviews with members of the group and with external people, e.g. customers, personal experiences, etc. For the sake of simplicity we assume that the ability and trustworthiness of a group can be represented as a single weight. Thus, an initial universe of discourse can be represented as:

```
UoD = {<V1,w1,g1>, <V2,w2,g2>, ... <Vk,wk,gk>}
      w1,w2,...,wk  weights assigned to the viewpoints
      g1,g2,...,gk goals
```

7.4.3 Importance Analysis

Importance analysis for this case study is roughly identical to the feasibility analysis of the solutions (i.e. production routes). To check the routes feasibility, each group proposes one or more solutions as 'production routes'. It is assumed that each group has evaluated their proposed solutions for their advantages and disadvantages. Because of the complexity of the problem there may be a number of possible solutions. The number of possible solutions needs to be reduced. Some of these solutions would be unfeasible on the basis of the overall limitation on technical grounds (e.g., if there is a deadline that all solutions have to respect). These solutions have to be eliminated. It should be clarified to the viewpoints why some solutions were declared unfeasible. Other solutions could be 'similar', according to an ad hoc metric that must be agreed with the customer or the experts before the process starts. In order to reduce the number of solutions further the remaining routes could be classified according to certain properties such as structure (S) and criterion (C): For example

```
S = (machines selected, operator, sequence)
C = (cost, relational stress, quality of the final product, marketability)
```

For example, the solutions can be classified as:

```
- similar structure and similar goals
- similar structure and different goals
- different structure and similar goals
- different structure and different goals
```

The result of importance analysis is the set of solutions deemed feasible.

7.4.4 Information Evaluation

Information evaluation estimates the acceptability of each feasible solution. Each production route is evaluated and assigned a belief (weight) - a global 'index of goodness' - in relation to their consistency with the technical, domain dependent knowledge and by checking the internal consistency of each pair route/experts according to the domain independent heuristics (eg, responsibility v. advantage).

7.4.5 Universe of Discourse Update

Once the routes have been received the groups' weights are modified using some information about the solutions. For example:

```
if all of a group's solutions have been eliminated or
have received a low index,  the weight should be decremented

if the solutions proposed by young experts are similar to
those of senior engineers,  their weights should be incremented

if a solution is such that it maximises only the goals
specific to the group's class, and low commitment is
accepted,  the weight should be decremented
```

The results of this and the previous phases are recorded as cases, with a case for each acceptable solution. For example:

```
case20:
        Route:_
```

```
   Structure:_
   Criterion:_
   Weight:_
   Viewpoint:_

Information Evaluation
   problem of Commitment:_
   Problem of Advantage:_
   Problem of Ability:_

Viewpoint Re-evaluation
  Weight:_
  Goal:_
```

7.4.6 Group Decision

The next step is concerned with the group decisional process. The process takes as input the set of cases and viewpoint models and selects a compromise. As described in Chapter 7, the Consensus Model [50] can be used at this stage:

- Each member of SET is asked to give their evaluation of the proposed solutions with respect to each criteria (i.e. with respect to the satisfaction of their private goal). The results of this phase of consultation is expressed in a matrix form. The elements of the matrix represents the 'linguistic performance' that the group have attributed to each solution with respect to each criteria, (that is, choosing a linguistic label represented by 'fuzzy numbers' in a term set V, the range of which is pre-defined. For example: V = (very low, low, medium, high, very high),

- taking into account the weights of the viewpoints, recorded in the viewpoint models, a consensus strategy is identified using a 'cost for changing opinion' (some form of commitment). There are two stages:

  1. identify candidate solutions for the discussion (eliminate those whose total value - as judged by the viewpoints - does not pass a fixed threshold),

2. evaluate the remaining solutions again after a discussion based on the advice of the consensus strategy, which in turn is based on the 'cost for changing opinion': this process may be repeated a number of times depending on various elements. The weights assigned to the viewpoints can also be changed as a result

- change the weights of the viewpoints.

If after a round of consultation there is not a Production Route that can acceptable as the final one, it is necessary to change the level of negotiation and change the Production Routes themselves. The viewpoints have to be told why a consensus has not been reached and on how to use this information as a starting point for the next stages.

Note that this example is only intended to show how a negotiation strategy can be, easily, integrated with the techniques of the Viewpoint Control Method. This is due to the flexibility of the method and the usefulness of the information it produces for the process of negotiation.

7.5 Summary

Using an example problem and two case studies this chapter has demonstrated the utility of the Viewpoint Control Method in the very early stages of requirements elicitation. The example problem is employed to illustrate the kind of problems this work is addressing and how the proposed approach can, with little difficulty, deal with situations Leite's approach cannot cope with. A particular situation the example sought to illustrate is when two statements (from the same or different viewpoints) are consistent but there is no evidence that they are correct with respect to the universe of discourse. Because the Viewpoint Control Method records who said what, the track-record of each source, and what evidence is provided to endorse what was said, it is able to use that evidence to judge the degree of correctness of individual pieces of information. The case studies sought to illustrate the following aspects of the proposed approach:

- the coordination of the different analysis techniques via the viewpoint models and the cases, thus the production of an all-important learning feed-back,
- the flexibility of the method. The method can be adapted to different situations in different domains. No order is imposed on performing the analysis tasks, e.g. iteration between conflict resolution and viewpoint re-evaluation, and no specific representation is imposed for expressing the facts,
- the important role human factors play even in technical decisions,
- the ability to detect inconsistencies, wrong information, and incompleteness as the requirements evolve,
- the exploitation of the correlation between inconsistency, incorrectness and incompleteness problems in order to make the maximum use of the information available,
- the utility of the method in supporting group decisions under uncertainty by recording information about the participants,
- the use of LOLITA as a natural language environment.

In the next Chapter the results and evidence presented in this Chapter are used to evaluate the Viewpoint Control Method from the point of view of its ability to distinguish between inconsistencies, wrong information and missing information, the extent to which the negotiation process is supported and in particular, how early the problems are detected.

Chapter 8

Evaluation of the Method

In this chapter the Viewpoint Control Method is evaluated against the criteria for success and compared with other methods. The method is evaluated from the point of view of its ability to distinguish between inconsistencies, wrong information and missing information, the extent to which the negotiation process is supported and in particular, how early the problems are detected.

8.1 Evaluation Against the Criteria for Success

The criteria for the success of this investigation have been met and are as follows:

1. the ability to detect problems earlier than the Leite method allows,
2. the ability to deal with incorrectness,
3. the ability to provide a better quality agenda,
4. the ability to deal with conflict resolution.

The Viewpoint Control Method is of an investigative nature and does not, therefore, assume the availability of the facts when a number of decisions and assumptions about the problem have already been made, as is the case with the existing, early validation methods such as Leite's. The method operates within a natural language environment, allowing it to acquire information from the viewpoints in a highly interactive mode. Individual pieces of information are evaluated as they are elicited.

A universe of discourse has been defined. It includes all the sources of information and their records, represented by the viewpoint models. The universe of discourse is domain-independent and is not assumed to be pre-defined. It is updated as the investigation progresses while the requirements are validated as they evolve.

As shown in the application of the method, the Viewpoint Control Method is able to distinguish between inconsistencies, wrong information, and missing information:

- Inconsistencies
  - the various parts of a statement (Certainty, Commitment, Advantage, etc) cannot consistently stay together.
  - a statement does not live up to the expectations of the viewpoint model
  - a statement at odds with another statement
- Wrong information
  - a statement is attributed a very low or nil belief
- Incompleteness
  - there is evidence of the need for more information that requires an enquiry. Although incompleteness is impossible to solve, it is possible to detect pointers to missing information.

The distinctive feature of the Viewpoint Control Method is its exploitation of the correlation between the problems of inconsistency, incompleteness, and incorrectness to form an opinion about a piece of information thus making the maximum use of the information available. The correlation is captured by the Evaluation-feedback loop. For example, an inconsistency may provoke an enquiry to find an explanation, and the enquiry may reveal evidence that may lead to the modification of the viewpoint model which in turn affects the decision on the degree of 'correctness' that can be attributed to the information.

Each piece of information treated by Viewpoint Control is identified with its source and each source with the information it contributed. The sources are represented by their viewpoint models (in the universe of discourse) and the information is represented by the cases, which record the results of the analyses and their rationales. The cases act as links between the universe of discourse and the information base. Thus they allow tracing information back to their originators. They also make it easier to look up the general results and problems of a previous stage of analysis to use them as a guide for further analysis and as a means to take an immediate decision if necessary. The heuristics that were used to reach a particular decision can also be recorded thus allowing the re-play of particular steps.

As illustrated by the Case Study 2 in Chapter 7, the Viewpoint Control Method supports the negotiation process in two aspects. First, the method supplies information about the participants. The information has been captured by maintaining viewpoint models. Second, because any conflict is considered by the method as inconsistency, the principles of importance analysis, information evaluation, enquiry, and viewpoint re-evaluation can be applied during a group decision. The result is an iterative process of fact-finding, communication, and fact-validation (see figure 1.1).

8.2 Strengths and Weaknesses

It has been shown that the Viewpoint Control Method achieves its objectives set out in chapter One. In addition, the method has a number of advantages some of which are a direct consequence of the objectives. For example, the method makes the maximum use of the information available by exploiting the correlation between the problems of inconsistency, incompleteness, and incorrectness.

Other strengths of the Viewpoint Control Approach, which may not be readily apparent, are recognised.

The method is highly flexible. The Viewpoint Control Method does not depend on the quality of the viewpoints that take part in the viewpoint resolution process. The viewpoints are not pre-defined. New viewpoints are included and their relative quality determined as the investigation progresses. This gives an analyst the freedom to regulate the space of the investigation. The method's domain and representation-independence adds to its flexibility.

The explicit use of human factors such as commitments will inevitably encourage responsible attitude of the participants, including the analysts and the decision makers [88]. Also, maintaining viewpoint models of the participants enables the analysts to learn from their mistakes.

The method distinguishes between inconsistencies, wrong information and missing information and is sensitive to previous analyses' results.

As a viewpoint resolution method the principles of the Viewpoint Control Method can be adapted to any software development activity. The second case study shows that a viewpoint can be equated with any expert of a particular domain and the same principles apply - analysts, designers, maintainers, etc. In addition, Viewpoint models recorded during requirements elicitation can be useful for later software development activities.

Requirements elicitation is still a 'mysterious' activity. This investigation provides a wealth of data to improve the understanding of this activity because it models many aspects of human behaviour

in acquiring information from human sources in an uncertain environment.

The integration of the different views is performed in parallel with their modelling rather than delayed until the views are 'final'. This has the advantage of allowing conflict resolution to be an integral part of the elicitation process, thus contributing to uncovering more information.

The following limitations of the Viewpoint Control method are recognised.

Although the method is domain-independent it is more useful for situations where most of the information comes from people and where there are few constraints imposed by the environment in which the software will operate. An example is the problem of defining the requirements for a decision support system to be used by a group of managers.

Although the natural language environment can provides a rich body of information about a particular event the Viewpoint Control's analyses are restricted to the event's external features (who said it, how it was said, how it relates to the existing events, etc.).

Considering every single piece of information is time consuming especially for large systems. This may force analysts to halt interacting with the viewpoints. A balance must be struck between maintaining interaction with the outside world and making the maximum use of the information available. Importance analysis is an attempt to force analysts to avoid unnecessary tasks. It is also allowed to set time limits or priorities to certain activities.

There is the risk that by using human factors, such as trustworthiness, some people may get alienated. The investigation may look like an inquisition.

Support tools are not implemented to carry out the heuristics and to help improving them.

8.3 Comparison with Other Methods

In the absence of a formal mapping between the acquired facts and the original intent, requirements validation research has been concentrating on improving the approximation between the gathered facts and the universe of discourse [72]. This includes detecting problems in the requirements and getting them resolved through negotiation. The difference between validation methods is the type and quality of the problems, that can be detected, and the level of support that can be provided to the conflict resolution process. Leite uses these criteria to evaluate his method (Viewpoint Analysis). Viewpoint Analysis is the closest rival to the Viewpoint Control Method.

The Viewpoint Control Method is superior to Viewpoint Analysis in the following aspects:

- Viewpoint Control validates facts earlier than Viewpoint Analysis. Viewpoint Control has pushed the boundaries of requirements validation upstream, towards fact-finding.
- Viewpoint Analysis does not actually deal with the correctness problem as claimed by the author (see section 3.2) whilst Viewpoint Control does detect wrong information
- Viewpoint Analysis does not support the conflict resolution process. As it has been shown, Viewpoint Control supports conflict resolution and has pushed the boundaries of requirements validation downstream, towards communication.

Another related area is viewpoint resolution, especially agent-oriented. To reiterate, a viewpoint resolution involves: identifying viewpoints, reasoning within and between viewpoints, and revising viewpoints. The differences between Viewpoint Control and the agent-oriented methods are:

- The agents' characteristics and relations (human factors) are treated more explicitly by Viewpoint Control
- None of the methods deal with conflict resolution as an explicit and integral part of the viewpoint resolution process. It has been shown how Viewpoint Control can easily accommodates

a negotiation strategy based on the fuzzy set theory. This is achieved through the capture of the 'roots' of the conflicts, represented by the viewpoint models.

- The existing methods do not provide procedures for revising viewpoints. Viewpoint Control achieved that through the viewpoint re-evaluation procedures and through the ability of re-palying lines of reasoning.

A similar approach to Viewpoint Control is the goal-directed strategy for requirements acquisition proposed by Fickas *al.* [52]. They suggested a set of heuristics as a basis for defining which system human agents should best perform which actions. Agents are assigned to actions depending on their ability, reliability and motivation. For example, no agent will be responsible for a goal in conflict with its private goal or if there are several candidate agents to perform an action an agent is selected so that the values of the ability and reliability are maximised.

From the validation perspective, the domain-specific approach [51, 102, 99] adopts a similar line to Viewpoint Control, namely:

- a universe of discourse is defined by the domain goals or domain cliches
- the universe of discourse is linked to the specification [102]
- the universe of discourse can be updated by revising the goals' attributes [51].
- conflict resolution is supported by appealing to the goals from which the conflict stems.

The approach of Viewpoint Control has also some similarities with the work reported in [58] with respect to the handling of inconsistencies in a multiple viewpoints specification. The two approaches share the idea of defining a reference against which inconsistency in the information is checked. A multiple viewpoints specification is seen as a database. The database has a context in which it operates. The context includes rules for using the database, integrity constraints, implicit assumptions, and information about the relations between the views making up the database. The context acts as a reference in which inconsistencies are checked. The type of actions that one needs to take in

the event of an inconsistency has been illustrated by two examples. The example of a government tax database, where an inconsistency may occur in a taxpayer's record. The inconsistency can invoke an investigation of that taxpayer. The other example is about the use of credit cards in a department store, where an inconsistency may occur on some account. In this case, they suggest, the store may take one of the series of actions such as writing off the amount owed, or leaving the discrepancy indefinitely, or invoking legal action. This is the type of actions that the Viewpoint Control Method promotes.

The difference with this work is that the Viewpoint Control Method does not only take action in the event of an inconsistency but also attempts to learn about the source of inconsistency, thus taking precautionary measures before other inconsistencies arise.

8.4 Summary

In this Chapter the Viewpoint Control Method is discussed in the context of very early validation in the process of requirements elicitation. The evaluation has shown that the method meets the criteria for success set in Chapter 1, and in particular the ability to distinguish between inconsistent, incorrect, and missing information. Also, the method exploits the correlation between inconsistencies, incompleteness and incorrectness problems, using a learning feed-back mechanism. The improvement of the proposed approach on Leite's method are apparent:

1. the support of the fact-finding process,
2. the support of view construction. To construct a view, Leite uses a shallow conceptual model. The model is composed of objects, agents, actions, attributes, and the hierarchies, is-a and parts-of. It has been shown that these entities and relations can be automatically extracted from a natural language text using LOLITA (L-Object_based-LITA),

3. the improvement of the quality of discrepancies, within and between viewpoints (e.g. incorrectness), that can be detected,

4. the support of the negotiation process.

The strengths and weaknesses of the method are also discussed. Finally the method is compared with other methods for requirements validation as well as methods for viewpoint resolution.

Chapter 9

Conclusions

9.1 The Main Achievements of the Research

The main achievement and result of this research is a method for the very early validation of requirements. The Viewpoint Control Method represents a novel approach to requirements elicitation. The principles of the method are drawn from the fields of uncertainty management, viewpoint resolution, and natural language engineering. The way in which these fileds are drawn together is novel, as is their application in the area of requirements engineering. In addition, the method has the following aspects:

1. The maintenance of **viewpoint models** and their use for assessing information from a **natural language** input.

2. The association of each participant with the information they contributed, and the recording of the analysis processes and their results, thus allowing the replay of those processes.

3. The treatment of conflict resolution as an explicit, and an integral part of the validation process.

4. The validation of natural language information.

5. The explicit use of human factors and relations in requirement engineering.

9.2 General Conclusions of the Research

The primary hypothesis of this thesis is that, independently of the quality of the viewpoints, the number of viewpoints, the language, the domain, it is possible to detect better quality discrepancies and to point out problems earlier than Leite's method allows. In evaluating the Viewpoint Control Method (see Chapter 8), it has been shown how the process of very early validation has been improved. There are a number of conclusions which can be inferred:

- learning about human sources and maintaining models of their behaviour contributes to the management of uncertainty with respect to the information coming from those sources,

- managing the uncertainty with respect to the information from multiple viewpoints improves the viewpoint resolution process. These conclusions support the view of the school of thought in Information Systems Theory that underlines the naivety of assuming competence or objectivity of each source of information.

- Asking the question: "Are we getting the right information?" is a more appropriate starting point for an early validation of requirements than asking the traditional question: "are we building the right product?" which is more appropriate for the later stages of requirements engineering.

- Human factors and relations play an important role in the very early validation of requirements. This role must be made explicit, in order to discount it from the decisions, while reducing the emotive impact on the participants.

- Conflict resolution should be an integral part of the validation process. There should be feedbacks between the conflict resolution and the information evaluation tasks, so the maximum use is made of the information available.

- The Application of AI techniques to SE problems is possible if severe restrictions are imposed on the application domain. For example, one may select a particular software engineering activity (e.g. early validation) within an existing software engineering method (e.g. CORE) in a particular paradigm (e.g. multiple viewpoints).

9.3 Relationship to the Wider Field

The work presented in this thesis has links with other software engineering and artificial intelligence topics:

- **Knowledge validation.** Chapter 7 has shown that the Viewpoint Control Method can deal with situations where the viewpoints are not necessarily requirements sources but can be any domain experts: maintainers, designers, etc. In fact, uncertainty, as defined in this thesis, is not limited to requirements elicitation [71]. Ole *et al.* [91] propose 'the validation of knowledge against the knowledge source' as an alternative to the traditional validation techniques in software systems development. But they gave no indication on how to achieve that. Viewpoint Control can be used as a knowledge validation technique.

- Systematic methods to guide the **Application of AI** techniques to software engineering. There have been very few suggestions in this area. Chapter 4 presents an informal process of applying an AI model (Source Control Mechanism) to a software engineering problem (requirements elicitation from multiple viewpoints). According to Boehm* there are the SE/AI perspective and the AI/SE. The SE/AI perspective is characterised as the selection of

*Remarks made at the *4th International Workshop of Software Specification and Design* (according to Freeman *et. al.* [11])

a restricted subset of SE problems and the adoption of ideas, techniques, and representations form AI to solve these problems in the context of SE. The AI/SE perspective is characterised by a reformulation of the SE processes in AI terms and an attempt to solve them entirely within AI. Boehm argued that SE/AI view offers some promise of feasible applications in the short term.

9.4 The Limitations of the Approach

The Viewpoint Control Method is domain-independent but it is more useful for situations where most of the information comes from people and where there are few constraints imposed by the environment in which the software will operate (a typical example is the requirements engineering needed to build decision support software to be used by a group of managers). Brackett [28] found that the fraction of requirements elicited from people increases as constraints on the software requirements process decrease.

Although the method is not dependent on the quality of the viewpoints, as is the case for Leite's Viewpoint Analysis, nor on the language used to express the information, its analyses are restricted to the external features of the information: who said it, how it was said, and how does it relate to the existing information. Finally, support tools have yet as not been implemented.

9.5 Suggestions for Future Research

9.5.1 Tool Support

Figure 6.2 can be seen as a high level design for a support tool. Apart from supporting the use of the method, the tools will be useful for experimenting with the method in order to improve it.

9.5.2 Validation by Generation

One of the techniques for specification validation is paraphrasing. Paraphrasing is the process of generating information in natural language from a formal specification so the specification can be checked for 'correctness'. The quality of information generated by paraphrasing depends on the semantics imposed by the formal specification language used. Generation in LOLITA operates on a rich representation (Conceptual Graphs), thus interpreting specification independently of the development method used. There exists a graphical interface (developed by Siemens Plessey) which allows exploration and manipulation of the semantic net (see Figure 4.3). Work is underway for the implementation of a dialogue analysis theory [65] and its incorporation in the system together with a stylistic generator.

9.5.3 Multiple Formalisms

Starting from the semantic net, in which redundancies and inconsistencies have already been solved or flagged for user's attention, requirements can be modelled using different modelling techniques to represent different aspects of the system (e.g. functional, informational, operational, etc). This is made easier by the fact that the requirements are represented in the semantic net independently of any development method.

But the central problem in using a real natural language system such as LOLITA for requirements modelling is not defining entities, or specifying connections, but rather of selecting only those needed in the application context such as patient-monitoring. The difficulty is that a requirements modelling scheme should provide a representation at the right level of description: neither too generic, nor too detailed. For example, consider the entity 'Roberto' (proper noun). From the net (see figure 4.3) it is known that 'Roberto' is : owner, bachelor, lecturer, etc. These are 'classes' in LOLITA. The class 'lecturer' might not be suitable for an application related to patient-monitoring, while it is essential for a system to assign timetable slots to courses and lecturers.

The solution requires the definition of an application 'locality': a similar problem is treated in [75], but is made more complex in this context by the fact some subclasses, even if they are richer in information under a logical point of view, are less relevant in the application context. It would seem that what is required is a semantic model of the application, to be used as a filter for the selection of the basic modelling units.

There is the problem of circularity in assuming a complete semantic model. However if the user specifies the general application type in a header (e.g. 'system for monitoring patients in an intensive care unit'), since these basic concepts are already defined in LOLITA, the header can be used as 'interest focus', from which to compute the 'semantic distance' of the various concepts. This approach could be used in conjunction with analysing the whole text looking for 'semantic clusters, and use that information to define the interest focus.

9.5.4 Specification Reuse

LOLITA has a powerful inference engine, based on an original form of Conceptual Graphs [109], which can perform (beyond standard inferences) multiple and frame inheritance, epistemic reasoning, causal reasoning, and reasoning by analogy. Added to 'template building', these capabilities would be very useful in performing 'fine grain' comparisons of specification 'templates', thus improving the quality of reusable components.

9.6 Summary

The contribution of this work is a method for the very early validation of requirements using a new approach to viewpoint resolution. The viewpoint resolution approach is centered around the idea of uncertainty management in a natural language environment. The Viewpoint Control Method involves maintaining models of the requirements sources. The source models are used to

evaluate information received from those sources. The models are then reassessed in the light of new evidence about the sources' behaviour. This chapter has presented the main achievements of the research, the general conclusions, the relationship with the wider field and some suggestions for future research.

Bibliography

[1] IEEE : **Standard Glossary of Software Engineering Terminology**, New York, IEEE, ANSI/IEEE Std 729-1983.

[2] Boehm, B.W., : **A Spiral Model of Software Development and Enhancement**, in *System and Software Requirements Engineering*, (eds.), Thayer, R.H. and Dorfman, M., pp. 513-527, 1990.

[3] **ACM Sigsoft Software Engineering Sympos. on Rapid Prototyping**, Columbia MD, ACM Sigsoft Software Engineering Notes, Vol. 7, No. 5, Dec. 1982, pp. 3-16.

[4] Abbot, R. : **Program Design by Informal English Description**, IEEE Transaction on Software Engineering 26(11), pp. 882-894, 1983.

[5] Adelson, B., and Soloway, E. : **The Role of Domain Experience in Software Design**, IEEE Trans. Soft. Eng. 1985, SE-11, (11), 1985.

[6] Ainsworth, M., Cruikshank, A.H., Wallis, P.J.L., and Groves, L.J. : **Viewpoint Specifications in Z**, Information and Software Technology, Vol. 36, No. 1, pp. 43-51, 1994.

[7] Alford, M. W. : **A Requirements Engineering Methodology for Real time Processing requirements**, IEEE Trans. Software Engineering, SE-3 (1), pp. 60-9, 1977.

[8] John, S., Anderson, S. and Fickas, S. : **A Proposed Perspective Shift: Viewing Specification Design as a Planning Problem**, ACM Sigsoft, Software Engineering Notes, Vol 14, No 3, pp. 177-184, May 1989.

[9] Alshawi *et al.* : **CLARE: A Contextual Reasoning And Cooperative Response Framework for the Core Language Engine**, Final report, SRI International, Cambridge Research Centre, Cambridge, England, 1992

[10] Appelt, D.E : **TELEGRAM : A Grammar Formalism for Language Planning.** In Proceedings of the 8th IJCAI, pp. 595-599. Karlsruhe, West Germany, 1983.

[11] Arango, G. and Freeman, P. : **Application of Artificial Intelligence to Software Specification and Design**, ACM Sigsoft, Software Engineering Notes, Vol 13, No 1, pp. 32-38, January 1988.

[12] Guillermo, A., Baxter, I. and Freeman, P. : **A Framework for Incremental Progress in the Application of Artificial Intelligence to Software Enginnering**, ACM Sigsoft, Software Engineering Notes, Vol 13, No 1, 1988, pp. 46-50.

[13] Bailin, S. : **An Object Oriented Requirements Specification Method**, Communication of the ACM **32**(5), 1989, pp. 608-623.

[14] Balzer, R. : **Transformational Implementation: An example**, IEEE Trans. on Software Engineering, Vol. 7, No. 1, Jan. 1981, pp. 3-13.

[15] Balzer, R., T. E. Cheatham, Jr., and Green : **Software Technology in the 1990's: using a new paradigm**, Computer, 16(11), 1983, 39-45.

[16] Balzer, R., : **A 15 Year Perspective on Automatic Programming**, IEEE Trans. on Software Engineering, Vol. SE-11, 1985.

[17] Balzer, R. and Goldman, N. : **Principles of Good Software Specification and their Implications for Specification Languages**, in *Software Specification Techniques*, edited by N. Gehani and D. McGettrick, pp. 25-39, 1986.

[18] Barnett, J., Knight, K., Mani, I., and Rich, E. : **Knowledge and Natural Language Processing.** Communications of the ACM, August 1990, v33, n8.

[19] Batini, C., Cenzerini, M., and Navathe, S.B. : **A Comparative Analysis of Methodologies for Database Scheme Integration**, ACM Computing Surveys, Vol 18, No 4, 1986.

[20] Belkhouche, B. and Kosma, J. : **Semantic Case Analysis of Informal Requirements**, in S. Brinkkemper & F. Harmsen, eds., *4th Workshop of the Next Generation of CASE tools (NGCT'93)*, Memoranda Informatica 93-32, Universiteit of Twente, The Netherlands, pp. 163-182.

[21] Bell, T. E., Bixler, D. C. and Dyer, M. E. : **An Extendable Approach to Computer aided Software Requirements Engineering**, IEEE Transactions on Software Engineering, SE-3 (1), 49-60, 1977.

[22] Boehm, B.W. : **Verifying and Validating Software Requirements and Design Specification**, IEEE Software, Vol 1, No 1, 1984, pp 75-88.

[23] Boehm, B.W, Gray, T.E., and Seewaldt, T. : **Prototyping Versus Specifying: a Multiproject Experiment**, Trans. on Software Eng., Vol. SE-10, No. 3, 1984, pp. 290-302.

[24] Bokma, A. : **A Source Modelling System and its Use for Uncertainty Management**, Ph.D. thesis, Department of Computer Science, University of Durham, 1994.

[25] Boland, R. : **Protocols of Interaction in the Design of Information Systems: An evaluation of the Role of Systems Analysts in Determining Information Systems Requirements**, Ph.D Thesis, Case Western, 1976.

[26] Booch, G. : **Object-oriented Development**, IEEE Transactions on Software Engineering **12**(2), 1986, pp. 211-221.

[27] Borgida, A., Greenspan, S., and Mylopoulos, J. : **Knowledge Representation as the Basis for Requirements Specifications**, Computer, April 1985, pp. 82-90.

[28] Brackett, J.W. : **Sofware Requirements**, in *Standards, Guidelines, and examples on System and Software Requirements Engineering*, M. Dorfman and R. H. Thayer (ed.), 1990.

[29] Brooks, P.F. : **No Silver Bullet - Essence and Accidents of Software Engineering,** IEEE Computer, Vol. 20, No.4, 1987, pp. 10-20.

[30] Brooks, P.F. : **People Are Our Most Important Product.** *The Educational Needs of the Software Community,* edited by Norman E. and Richard E. Fairley, Springer-Verlag, 1987.

[31] Brown, R.R. : **The Techniques and Practices of Structured Design A La Constantine,** *Conf. Notes, Infotech State of the Art Conf. on Structured Design,* Feb. 1977, pp. 75-97.

[32] Burstall, R.M. and Goguen, J.A. : **Putting Theories Together to Make a Specification,** *Proc. Fifth Joint International Conference on Artificial Intelligence,* Cambridge Mass., pp. 1045-1058, 1977.

[33] Cauvet, C., Proix, C. and Rolland, C. :**ALECSI: An Expert System for Requirements Engineering,** in Anderson, Bubenko & Solvberg, eds, *3rd International Conference on Advanced Information Systems Engineering (CAiSE'91),* LNCS 498, Springer Verlag, Trondheim, Norway, pp. 31-49.

[34] Chen, L. and Avizienis, A. : **N-version Programming: a Fault-tolerance Approach to Reliability of Software Operation,** in *8th Ann. Int. Conf. on Fault Tolerance Computing,* Toulouse, France, 1978, pp. 3-9.

[35] **Basic Training in Systems Analysis,** second edition, A. Daniels and D. Yeates (Eds.), National Computing Centre, 1971.

[36] Davis, M.A. : **The Analysis and Specification of Systems and Software Requirements,** in *System and Software Requirements Engineering,* Thayer R.H. and Dorfman M. (ed.), 1990.

[37] Davis, A.M., Bersoff, E.H., and Comer, E.R. :**A Strategy for Comparing Alternative Software Development Life Cycle Models,** IEEE Transaction on Software Engineering, 1988, pp. 1453-1461.

[38] Davis, G. : **Strategies for Information Requirements Determination,** IBM System Journal, Vol. 21, Part 1, 1982, pp. 4-38.

[39] DeMarco, T. : **Structured Analysis and Systems Specification**, Yourdon Press, 1979.

[40] Dewar, R., A. Grand, S. Liu, and J. Schwartz : **Programming Refinement, as Exemplified by the SETL Representation Sublanguage**, ACM Trans. on Programming Languages and Systems, Vol. 1 No. 1, July 1979, pp. 27-49.

[41] Doyle, J. : **Truth Maintenance Systems for Problem Solving**, AI-TR-419, AI Labs, MIT, 1978.

[42] Dubois, E. : **A Logic of Action for Supporting Goal-oriented Elaborations of Requirements**, ACM SIGSOFT Engineering Notes, Volume 14, Number 3, pp. 160-168, May 1989.

[43] Efstathiou, J.H. : **A Practical Development of Multi-attribute Decision Making using Fuzzy Set Theory**, Ph.D thesis, Department of Computing, University of Durham, 1979.

[44] Dunn, L. and Orlowska, M. : **A Natural Language Interpreter for the Construction of Conceptual Schemas**, in B. Steinholtz, A. Solvberg and L. Bergman, eds, *2nd Nordic Conference on Advanced Information Systems Engineering (CAiSE'90)*, LNCS 436, Springer Verlag, Stockholm, Sweden.

[45] Easterbrook, S. : **Elicitation of Requirements from Multiple Perspectives**, Ph.D. thesis, Department of Computing, Imperial College of Science, Technology, & Medicine, University of London, 1991.

[46] Elam, J.J., Diane B. Walz, D.B., Krasner, H., and Curtis, B. : **A Methodology for Studying Software Design Teams: An Investigation of Conflict Behaviours in the Requirements definition Phase**, *2nd Workshop on Empirical Studies on Programmers*, 1987, pp. 83-99.

[47] Fawcett, R. and Davies, B. : **Monologue As A Turn In Dialogue.** in R.Dale E.Hovy D.Rosner O.Stock *Aspects of Automated NLG*, LNAI, Vol. 587 Springer-Verlag, 1992.

[48] Feather, M.S. : **Language Support for the Specification and Development of Composite Systems**, Information Science Institute, Marina Del Rey, Ca., 1985.

[49] Feather, M.S. : **Detecting Interference when Merging Specification Evolution**, ACM Sigsoft, Software Engineering Notes, Vol 14, No 3, 1989, pp. 169-176.

[50] Fedrizzi, M., Mich, L., and Gaio, L. : **A Fuzzy Logic-Based Model for Consensus Reaching in Group Decision Support**, in *Proc. of International Conference on Software Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'92)*, Spain, pp 301-304, 1992.

[51] Fickas, S. and Nagarajan, P. : **Critiquing Software Specification**, IEEE Software, 1988, pp. 37-46.

[52] Dardenne, A., Fickas, S., and van Lamsweerde, A. : **Goal-directed Requirements acquisition**, *Six International Workshop of Specification and Design*, IEE Computer Society, Italy, 1991.

[53] Fillmore, C. : **The Case for Case**, jn E.Bach & R. Harms, eds, *Universal Linguistics*, Rinehart and Winston, Chicago, 1968, pp. 1-90.

[54] Finkelstein, A. and Potts, C. : **Evaluation of Existing Requirements Extraction Strategies**, Alvey FOREST project, 1985.

[55] Finkelstein, A. : **Reuse of Formatted Requirements Specifications**, Software Engineering Journal, 1988, pp. 186-197.

[56] Finkelstein, A. and Fuks, H. : **Multi-party Specification**, ACM SIGSOFT Engineering Notes, Volume 14, Number 3, 1989pp 185-195.

[57] Finkelstein, A., Kramer, J., and Goedicke, M. : **Viewpoint Oriented Software Development**, *3rd International Workshop on Software Engineering and its Applications*, Toulouse, France, (IEEE Computer Society), 1990, pp. 337-351.

[58] Finkelstein, A., Grabbay, D., Hunter, A., Kramer, J., and Nuslibeh, B. : **Inconsistency Handling in Multiple-perspective Specifications**, in *4th European Software Engineering Conference (ESEC'93)*, I. Sommerville and P. Manfred, eds., LNCS 717, Springer-Verlag, Garmisch-Partenkirchen, Germany, 1993, pp. 84-89.

[59] Flynn, D.J., Layzell, P.J., and P. Loucopoulos, P. : **Assisting the Analyst - The aims and approaches of the Analyst Assist project**, in *Software Engineering 86*, edited by D. Barnas and P. Brown.

[60] Freeman, P. : **Requirements Analysis and Specification: the first step**, in *Tutorial on Software Design Techniques*, IEEE Comp. Society, 1979.

[61] Garigliano, R., Morgan, R.G., and LOLITA Group : **The LOLITA Project: the First Eight Years**, under negotiation with Lawrence Earlbaun, UK, 1993.

[62] Heidorn, G. : **Automatic Programming Through Natural Language Dialogue: A Survey**, in *Readings in Artificial Intelligence and Software Engineering*, Morgan Kaufmann Publishers Inc. , Los Altos, CA, 1986, pp. 203-214.

[63] Jackson, M.A. : **Jackson System Development**, Academic Press, 1975.

[64] Jeremaes, P., Khosla, S., and Maibaum, T.S.E. : **A Modal (Action) Logic for requirements specification**, in *Software Engineering 86*, edited by D. Barnes and P. Brown, pp. 278-294.

[65] Jones, C.E. and Garigliano, R. : **Dialogue Analysis and Generation: a Theory for Modelling Natural Language English Dialogue**, in *Proceedings of EUROSPEECH '93* Vol. 12, p. 951, Berlin, September 1993.

[66] Bennett, K.H. : **The Post Graduate Handbook**, Department of Computer Science, University of Durham (SECS), 1993.

[67] Kay, M. : **Functional Grammar**, in *Proceedings of the Fifth Annual Meeting of the Berkeley Linguistic Society*, 1979.

[68] Kotonya, G. and Sommerville, I. : **Viewpoints for Requirements Definition**, Software Engineering Journal, 1992, pp. 375-387.

[69] Kramer, J., Chinnick, S., and Finkelstein, A. : **TARA: Tool Assisted Requirements Analysis**, Final Report September 1987, Research Report 87/18, Imperial College, London and Systems Designers plc, Camberley, Surrey, U.K.

[70] Wing, L. : **An issue-Based Framework for Requirements Elicitation, Description and Validation**, Technical report No. 92/02, Department of Computing, King's college London (University of London), 1992.

[71] Lehman, M.M. : **Uncertainty in Computer Applications is Certain - Software Engineering as a Control**, Imperial College Research Report DOC 90/2.

[72] Leite, J. : **Viewpoint Resolution in Requirements Elicitation**, Ph.D thesis, Department of Computer Science, University of California, Ivrine, 1988.

[73] Leite, J. : **Viewpoint Analysis: A Case Study**, ACM SIGSOFT Enginering Notes, Volume 14, Number 3, 1989, pp 111-123.

[74] Leite, J. and Freeman, P. : **Requirements Validation Through Viewpoint Resolution**, IEEE transactions on Software Engineering Vol. 17, No. 12, 1991.

[75] Long, D. and Garigliano, R. : **Reasoning by Analogy and Causality (A model and application)**, Ellis Horwood, 1994.

[76] London, K.R. : **The People Side of Systems: The Human Aspects of Computer Systems**, McGRAW-HILL Book Company (UK) Limited, 1976.

[77] Loucopoulos, P. and Champion, R.E.M. : **Knowledge-based Support for Requirements Engineering**, Information and Software Technology, pp. 115-133, vol. 31 no. 3, April 1989.

[78] Loucopoulos, P. and Champion, R.E.M. : **Concept Acquisition and Analysis for Requirements Specification**, Software Engineering Journal, Vol. 5, No. 2, pp.116-124, March 1990.

[79] Lundeberg, M. : **An Approach for Involving the Users in the Specification of Information systems**, in *Software Design Techniques*, Freeman and Wasserman (eds.), IEEE Computer Society, CA, 1980.

[80] Mann, W.C : **An overview of the Penman Text Generation System**, in *Proceedings of the National Conference on Artificial Intelligence*, 1983, pp. 261-265

[81] Meyers, S. : **Difficulties in Integrating Multiview Development Systems**, IEEE Software, Vol. 8, No. 1, 1991, pp. 49-57.

[82] McKeown, K.R : **Text Generation**, C.U.P, 1985

[83] Meyer, B. : **On Formalism in Specification**, IEEE Software, Vol.2, No. 1, pp. 6-26, IEEE, New York, jan. 1984.

[84] Mich, L. and Garigliano, R. : **Negotiation and Conflict Resolution in Production Engineering Through Source Control**, Technical report, Department of Computer Science, University of Durham, 1994.

[85] Kanth, M. and Harandi, M. : **Analogical Approach to Specification Derivation**, ACM SIGSOFT, Software Engineering Notes, Vol. 14, No. 3, 1989, pp. 203-210.

[86] Morrison, N. : **Communicating with Users during System Development**, Vol 30, No 5, pp. 295-298, June 1988.

[87] Mullery, G. : **CORE - A method for COntrolled Requirements Expression**, *Proc. of Fourth IEEE Int. Conf. on Soft. Eng.*, Munich, W. Germany, 1979.

[88] Mullery, P. : **Acquisition-Environment**, in *Distributed systems: Methods and tools for specification*, Spriner-Verlag, 1985.

[89] McDonald, D.D.and Meteer, M.M : **From Water to Wine : Generating Natural Language Text from Today's Applications Programs.** In *Proceedings of the Second Conference on Applied Natural Language Processing*, Austin, Texas, 1988, pp. 41-48.

[90] Niskier, C., Maibaum, T., and Schawbe, D. : **A Look Through PRISMA: Towards Pluralistic Knowledge-based Environments for Software Specification Acquisition**, ACM Sigsoft, Software Engineering Notes, Vol 14, No 3, 1989, pp. 128-136.

[91] Ole, J.M. and Slinlef, D. :**Knowledge Validation: Principles and Practice**, IEEE Expert, Vol. 8, No. 3, 1993, pp. 62-68.

[92] Patten, T. : **Systemic Text Generation as Problem Solving**, Cambridge University Press, 1988.

[93] Philips, E.M and Pugh, D.S. : **How to Get a Ph.D. : Managing the Peaks and Troughs of Research**, Open University Press, 1987.

[94] Pitrat, C. : **An Artificial Intelligence Approach to Understanding Natural Language**, North Oxford Academic, 1988.

[95] Potts, C. : **Seven (Plus or Minus Two) Challenges for Requirements Research**, *Six International Workshop of Specification and Design*, IEE Computer Society, Italy, 1991.

[96] Raiffa, H. : **The Art and Science of Negotiation**, Harvard University Press, 1982.

[97] Ramamoorthy, C. V., Miguel, L., and Shim, Y.C. : **On Issues in Software Engineering and Artificial Intelligence**, IJSEKE, Vol. 1, No. 1, March 1991, pp. 9-20.

[98] Rich, C. and Waters, R. (ed.) : **Introduction**, in *Readings in Artificial Intelligence and Software Engineering*, Morgan Kaufmann Publishers Inc. , Los Altos, CA, 1986.

[99] Rich, C., Waters, R., and Reubenstein, H. : **Towards a Requirements Apprentice**, *Fourth Int. Workshop on Software Specification and Design*, IEEE Computer Society, 1987, pp. 78-86.

[100] Rich, C. and Waters, R. : **The Programmer's Apprentice**, ACM Press, New york, 1990.

[101] Rego, H. and Lima, J. : **A Tool for Automating Facts Analysis**, in K.Spurr & P. layzell, eds, *CASE on Trial*, John Wiley, England, 1990, pp. 57-80.

[102] Robinson, W.N. : **Integrating Multiple Specifications using Domain Goals**, ACM SIGSOFT Engineering Notes, Volume 14, Number 3, 1989, pp 219-226.

[103] Ross, D.T. : **Structured Analysis (SA) : a Language for Communicating Ideas**, IEEE Trans. on Soft. Eng., SE-3, (1), 1977, pp. 16-33.

[104] Ross, D.T. and Schoman, K.R. : **Structured Analysis for Requirements Definition**, IEEE Trans. Software Engineering, Vol. SE-3, No. 1, 1977, pp. 16-34.

[105] Saeki, M., Horai, H., and Enomto, H. : **Software Development Process from Natural Language specification**, in *International Conference on Software Engineering (ICSE'89)*, ACM, Pitsburgh, 1989, pp. 64-73.

[106] Short, S. : **Semantic Analysis: Literature Survey**, AI Research group, SECS, University of Durham.

[107] Smith, M.H., Garigliano, R., and Morgan, R.G. : **Generation in the LOLITA System: An Engineering approach**. Submitted to the *Seventh International Workshop on Natural Language Generation*, Maine, 1994.

[108] Soloway, E. : **What to Do Next: Meeting the Challenge of Programming-in-the-large**. In E. Soloway and S. Iyengar, (eds.), *Empirical Studies of Programmers*, Norwood, NJ: Ablex, 1986, pp. 263-268.

[109] Sowa : **Conceptual Structures**, Addison Wesley, 1983.

[110] Smith and Davis : **Frameworks for Cooperation in Distributed Problem Solving**, IEEE Trans. Systems, Man & Cybernetics, Vol. 11, No. 1, 1981, pp. 61-69.

[111] Sycara, K. : **Resolving Adversarial Conflicts: an Approach Integrating Casae-based and Analytic Methods**. Ph.D thesis, Georgia Institute of Technology, 1987.

[112] Alejandro, T. : **On Mixing Formal Specification Styles**, *Fourth International Workshop on Software Specification and Design*, Monterey, California, USA, 1987, pp. 28-33.

[113] Thimbleby, H. : **Delaying Commitment**, IEEE Software, Vol. 5, No. 3, 1988, pp. 78-86.

[114] Vilatari, N.P., and Dickson, G.W. : **Problem-solving for Effective Systems Analysis: an Experimental Exploration**, Communication of the ACM, 1983, 26, (11).

[115] Wing, J.M. : **A Study of 12 Specifications of the Library Problem**, IEEE Software, pp. 66-76, July 1988.

[116] Winograd, T. and Flores, F. : **Understanding Computers and Cognition: A New Foundation For Design**, Ablex Publishing Corp., 1986.

[117] Zave, P. : **A Compositional Approach to Multi-Paradigm Programming**, IEEE Software, Vol. 6, No. 5, 1989, pp. 15-25.