

Handling Conflict Between Domain Descriptions With Computer-Supported Negotiation

STEVE EASTERBROOK

School of Cognitive and Computing Sciences,

University of Sussex,

Falmer, Brighton, BN1 9QH, UK.

<e-mail: Easterbrook@cogs.susx.ac.uk>

Conflict is an inevitable part of both knowledge elicitation and system design. People will disagree over how to interpret features of the application domain, what the requirements for a new system are, and how to meet those requirements. Conventional systems analysis techniques avoid such conflicts, making any resolution untraceable and adding to the communication problems. This paper surveys a number of fields which have addressed the problems of conflict resolution. A model of computer-supported negotiation is presented which can be used to address conflicts in systems analysis directly. The model begins with an exploratory phase, in which the conflict is broken down into its components, eliciting the issues which underlie disagreements and criteria to measure their satisfaction. A set of options for possible resolutions are generated using design techniques. Finally, these options are compared to the original issues, and evaluated according to the criteria associated with the issues. The model emphasizes communication, and encourages investigation of other viewpoints. The model has been used to develop a system called Synoptic, which provides a set of tools to support the exploration of conflicts.

1. INTRODUCTION

This paper discusses the occurrence of conflict in the software engineering process, and how it might be handled. In the computing literature, mention of the need to handle conflict is rare, which is perhaps surprising given the importance attached to it in the social sciences. For many years it has been recognised in management science and sociology that conflict is an inevitable feature of group interaction, to be harnessed for its positive aspects, rather than suppressed [Robbins 1989], [Deutsch 1973], [Strauss 1978]. Some recent software engineering research has identified conflict as an issue [Curtis, Krasner & Iscoe 1988], [Anderson & Fickas 1989], [Feather 1989], [Robinson 1990], although as yet little progress has been made towards understanding how such conflict might be handled.

Examples of many different types of conflict abound in the literature on organisational psychology. For example, Robbins [1974] describes a newly elected city manager who has promised an immediate improvement in rubbish collection. After several months the citizens complained that there was no improvement. On investigation it turned out that the citizens regarded "improved service" to mean more frequent collection, whereas the city manager had meant earlier, quieter and more economical collection. Thomas [1976] illustrates a different type of conflict, in an example of a manager who wanted his staff to adopt a new form for their weekly reports, whilst they preferred the old form. Again, investigation revealed the underlying cause: the staff did not feel they had the time required to complete the new form, while the manager needed the extra information which was not on the old form. Both these examples are typical of organisational conflict, and it is not difficult to see how similar conflicts arise during the introduction of a new software system. Similarly, the group developing the software may be subject to such conflicts during the development process.

In this paper, the term conflict is used in its widest sense, to cover any interference in one party's activities, needs or goals, caused by the activities of another party. Conflict can be characterised as disagreement among the originators of the requirements and this disagreement may lead to inconsistencies in the specification. However, disagreements do not always lead to inconsistency, and inconsistencies do not always indicate the presence of conflict.

Typical software engineering methodologies do not explicitly address the resolution of conflict. Such methodologies are geared towards maintaining consistency and so do not allow conflicts to be expressed, let alone constructively resolved. Indeed, we could characterise existing software methodologies as conflict avoidance, in that they prescribe particular approaches, which assist software practitioners in breaking problems down and resolving design decisions in particular ways. In this way uncertainty is reduced by the provision of the collective wisdom embodied in the methodology [Lehman 1990].

While this approach helps to avoid conflicts during development, it does not help to deal with conflicting requirements. Most methodologies require a single consistent specification as a basis for a coherent design, which means that conflicts are suppressed when the specification is written. Formal methods do not necessarily help, even though formal languages are intended to prevent ambiguity and inconsistency [Finkelstein, Finkelstein & Maibaum 1990]. The ability to formally reason with specifications is a huge step towards detecting the presence of conflict, but carries the implication that inconsistencies are errors which must be eliminated. Methods developed to support formal specification reflect this philosophy, and miss the chance to explore conflict.

If much of software engineering is geared towards conflict avoidance, this in itself may not be a problem, for two reasons. Firstly, conflict may not be as extensive in requirements engineering as has been suggested, so that studying it might only help in exceptional cases. Secondly, avoidance is a valid way of tackling conflict, especially where it prevents energy being wasted on fruitless confrontation. In this case we need to show that the conflict is important enough not to be avoided, and that a project can be enhanced by handling conflict in more direct ways.

We argue in this paper that conflict is extensive in most real domains, and that avoidance is not a satisfactory approach to handling conflicting requirements. Examples are given which show that conflict is important enough not to be suppressed, and that a specification can be enhanced by handling conflict in more direct ways.

1.1. Sources of Conflict

It has been demonstrated that conflict is common in group interactions [Robbins 1989]. We can therefore assume that any application domain involving more than one person will be subject to typical group conflicts. While it might be argued that a design process with a single goal, perceived in the same way by all participants, might be free of conflict, few real design processes are of this nature. This immediately suggests two possible sources of conflict in a real-world design process: conflict between the participants' perceptions of the domain, and conflict between the many goals of a design.

The extent of conflict in software engineering has recently been revealed by a major field study of software projects [Curtis, Krasner & Iscoe 1988]. Focussing on the behavioural aspects of software design, this study identified three major problem areas: the thin spread of application domain knowledge; fluctuating and conflicting requirements; and breakdowns in communication and co-ordination. Each of these problem areas is a source of conflict, and each depends crucially

on communication between participants as a basis for any solution. A good conflict resolution approach necessarily emphasises communication between parties.

Conflicting and fluctuating requirements have many causes, from change in the organisational setting and business milieu, to the fact that the software will be used by different people with different goals and different needs. Handling constant change in requirements (which has been termed *requirements maintenance* [Finkelstein, Goedicke *et. al.* 1989]) requires an evolutionary approach that must be based on accurate capture of rationales and process information.

Unless the application domain with which the software deals is free of conflict, then the resulting software must incorporate this conflict. For small programs, the domain can be restricted until the conflict is excluded. For any large scale software, this is not practical. When the application knowledge is spread over many people, there is likely to be much disagreement between them, and fitting together the many contributions will inevitably lead to inconsistency.

Even if a domain appears to be free of conflict, quite often there will be areas in which there are different ways of looking at things. While such perspectives may not be fundamentally incompatible, they are likely to appear inconsistent, and so lead to conflict. Even if participants are describing essentially the same concepts, the style in which these are described may vary: even formal representation schemes allow enough variation in style so that there may be many different ways of saying the same thing.

Other sources of conflict include: conflicts between suggested solution components; conflicts between stated constraints; conflicts between perceived needs; conflicts in resource usage; and discrepancies between evaluations of priority.

1.2. Conflict Resolution

There are a number of problems with any method which avoids conflict. Where conflicts do occur, they are likely to get suppressed, because there is no means of expressing them within the framework. It is possible that these conflicts will remain suppressed, leading to dissatisfaction with the specification and the process that led to it. Often, a single perspective will be adopted as the basis for the specification at the cost of any alternative perspectives.

If these conflicts are eventually resolved, the resolution must be carried out outside the framework of the method and consequently is likely to be carried out at an inappropriate time, using an undesirable means. In addition, resolution thus achieved is untraceable, making rationales invalid, and decisions irreproducible.

Suppression of conflict will have serious effects on the remainder of the software development process. In the worst case, suppressed conflicts may lead to the breakdown of the requirements process, or the withdrawal of participants. Failure to recognise conflict between the perspectives of the participants will cause confusion during the requirements phase, which will then continue throughout the lifecycle. The participants' understanding of the specification will differ, leading to further misunderstandings during design and implementation.

These problems make a study of conflict resolution desirable. We can draw on many fields which have addressed some or other aspect of the problem, in order to understand the processes involved. The next section surveys research on conflict in relevant fields, discussing in each case how the work might apply to software engineering. The third section synthesizes a model of conflict resolution, using negotiation, which is then used as a basis for computer support.

2. SURVEY OF RELEVANT FIELDS

This section reviews a number of fields in which conflict and its resolution are studied. These fields fall roughly into two traditions – the mathematical and the behavioural – which approach conflict in very different ways. The former are concerned with numerical models of the processes of bargaining and decision making, while the latter are concerned with social responses to conflict and with how people approach negotiation, particularly in the context of organisational behaviour. We also survey work related to conflict resolution within fields of computer science and software engineering.

2.1. Terminology

Before we survey other fields, it is useful to introduce a suitable terminology. We will talk about conflict between *parties*, as conflict may occur between individuals, groups, organisations, or even between different roles played by one person. Similarly, when discussing conflict resolution, we will refer to *participants* of the resolution process, to cover a similar diversity. Not all parties to a conflict need necessarily be participants in its resolution.

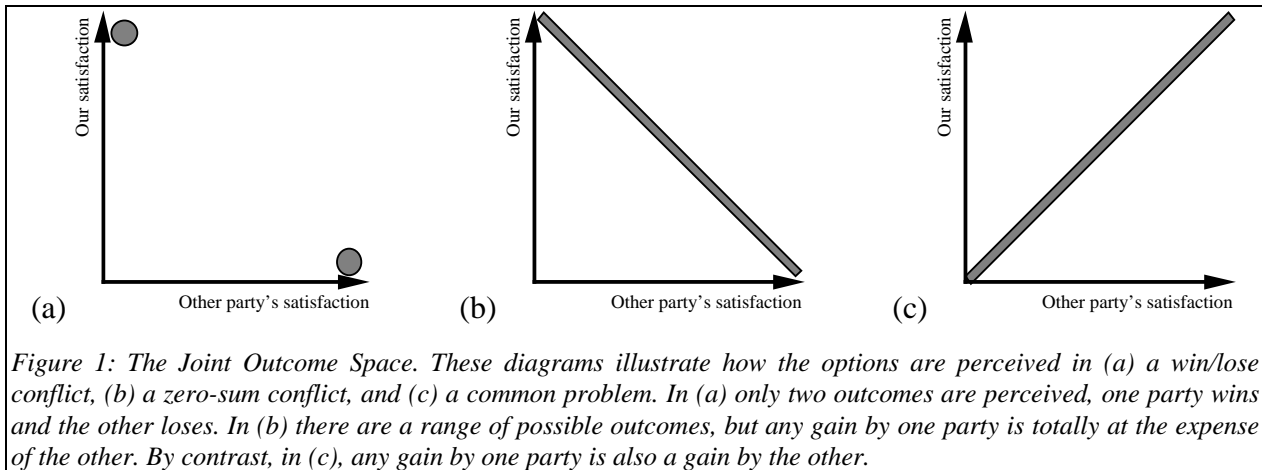
The approach used to settle a conflict is a *Resolution Method*. Methods include negotiation, competition, arbitration, coercion, and education [Strauss 1978]. Not all conflicts need a resolution method, as not all conflicts need to be resolved. Three broad types of resolution method can be distinguished: *Co-operative* (or collaborative) methods, which include negotiation and education; *Competitive* methods, which include combat, coercion and competition; and *Third Party* methods, which include arbitration and appeals to authority.

Negotiation is a collaborative approach to resolving conflict by exploration of the range of possibilities. It is characterised by the participants attempting to find a settlement which satisfies all parties as much as possible. Such an approach has been variously termed *integrative behaviour* or *constructive negotiation* (to distinguish it from *distributive*, or *competitive* negotiation). This definition of negotiation is not universal. Authors such as De Bono [1985] restrict negotiation to its distributive variety, implying a process of bidding and concession-making, and so attack it as being inferior to an integrative approach. We prefer to give negotiation its broader definition, and call the concession-making process *Bargaining*.

There are other collaborative methods than negotiation. Some conflicts might be resolved by education, where the participants gain a better understanding of the problem, or simply learn about each other's viewpoint. Another important technique is to reformulate the problem, so that it disappears, or becomes unimportant.

In contrast to negotiation, *Competition* concentrates on achieving maximum satisfaction for a participant, without regard for the degree of satisfaction of other parties. However, a competitive approach is not necessarily hostile. An extreme form of competition is when all gains by one party are at the expense of others, which, in game theory, is termed a zero-sum game.

Third Party Resolution covers any situation where participants are unable to resolve a conflict between themselves, and so have to appeal to an outside source, whether this be the rule-book, a figure of authority, or the toss of a coin. Such a situation can occur with the breakdown of either negotiation or competition as resolution methods. There are two types of third party resolution: those in which the cases presented by each participant are taken into account, which we might term *judicial*; and those where a decision is determined arbitrarily (e.g. tossing a coin), or by



factors other than the cases presented (e.g. by the relative status of the participants), which we might term *extra-judicial*.

Bidding and *Bargaining* are phases of the resolution process. Bidding is where participants state their desired terms for the settlement, often with an indication of the relative importance of them, as a basis for bargaining. Bidding takes place in some form or other in most resolution methods, as participants must present their side, although in methods such as coercion, the bidding might be one-sided and implicit. A *position* is the set of terms that a participant commits itself to by making a bid. In bargaining, participants search for a satisfactory integration of bids. In the simplest case this involves a converging sequence of bid and counter-bid, while at the other extreme, participants seek to blend complex bids together. Note that the description of the outcome as *satisfactory* depends on your viewpoint. However, bargaining usually results in a compromise, whereas true constructive negotiation seeks to develop a new solution which fully satisfies all participants.

2.2. Mathematical and Economic Models

Decision theory is a prescriptive approach to analysis of a set of pre-specified alternatives. The interesting problems are concerned with resolving multiple conflicting objectives [Keeney & Raiffa 1976]. Decision theory assumes a single entity is making a choice, in contrast to conflict where there is more than one entity, each with a different perspective. Decision theory has a role in conflict resolution in helping participants to evaluate bids, to justify such evaluations, and to persuade the other participant(s) that a solution is satisfactory.

Bargaining theory is an attempt to produce descriptive models of bargaining processes, and is especially concerned with commerce and politics. Patchen [1970] surveys models of bargaining theory and notes that the more complete models include wider concerns than bids and outcomes, including how participants influence each other's behaviour, and factors such as the cost of various actions and the cost of delaying agreement. Bargaining theory frequently makes use of the *joint outcome space* [Thomas 1976] as a tool for illustrating how the parties perceive the options in a conflict (figure 1). Note that there may be possibilities not perceived by the participants, which provide better resolutions. Bargaining theory does not indicate how these might be found, concentrating instead on the process of bidding and counter-bidding.

Game Theory is defined as the theory of rational decision in conflict situations [Rapoport 1974]. Participants are regarded as players, and game theory examines the strategies used by the players in the process of trying to achieve particular outcomes. In contrast to the joint outcome space, game theory often makes use of the payoff matrix (figure 2). This reflects the assumption that the set of outcomes is known (though not necessarily finite), and that associated with each outcome is a calculable payoff for each player. Limitations of game theory include the restricted sets of available actions, and the assumption that the payoffs for any action are known with certainty by all players. However, game theory does produce some useful information about the kinds of strategy that can be used to induce co-operation and how various strategies pay off for the players [Axelrod 1984].

Group Decision Making is the normative study of how individual preferences can be combined into a group decision. Luce and Raiffa [1957] defined the problem as that of finding a method, or *welfare function* for combining individual preference rankings into a social preference, which satisfies properties such as fairness and representativeness. Work on group decision making extends decision theory to cope with more than one decision maker, but still suffers from the assumption that all the options are known. In design, we are usually concerned with creating new options, rather than selecting among existing ones.

2.3. Behavioural Models

Conflict Theory studies the conflicting pressures of various groups in society, recognising that there are many different groups in society with different goals, and that conflict is a frequent occurrence. Strauss [1978] points out that the majority of conflicts are resolved by co-operative means, often unconsciously, and yet despite this, very little attention is paid to these co-operative mechanisms. Deutsch [1973] begins to examine the nature of conflict, and gives a list of issues involved in conflicts:

- Control over resources
- Preferences and nuisances, where the tastes or activities of one party impinge upon another.
- Values (“what should be”), where there is a claim that a value, or set of values should dominate
- Beliefs (“what is”), when there is a dispute over facts, information, reality etc.
- The nature of the relationship between the two parties

This list closely follows those suggested by Robbins [1989], who adds that communication problems are a major cause of what he terms *pseudo-conflicts*, and De Bono [1985], who notes that the usual thinking styles encourage people to be contrary.

One of the major concerns of *Organisational Psychology* is team-work within organisations, and how communication and co-ordination of teams can be effected. Early work tended to assume all

		Prisoner B	
		<i>Not Confess</i>	<i>Confess</i>
Prisoner A	<i>Not Confess</i>	1 year each	10 years for A and 3 months for B
	<i>Confess</i>	3 months for A and 10 years for B	8 years each

Figure 2: The payoff matrix for the prisoner’s dilemma. Each player must decide, in isolation from the other, whether to confess to a crime that the judge is sure they both committed. By confessing each will implicate the other, and their joint best strategy is for both to keep quiet.

conflict was undesirable, and so should be eliminated, although empirical studies in the last few decades have demonstrated that conflict is an inevitable feature of group interaction. Moreover, Robbins [1974], among others, has advocated that conflict management include not just resolution of conflict, but stimulation of conflict too. This is a result of observations that conflict has a useful role in organisations, in providing a stimulus to innovation, as it involves questioning and evaluating received wisdom. It is also a major weapon against stagnation and resistance to change.

In his model of the stages in the development of conflict in group behaviour, Robbins [1989] groups the conditions which allow conflict to arise as follows:

- Communicational, including: insufficient exchange of information; noise; and the semantic differences that arise from selective perception and difference of background.
- Structural, including the goal compatibility of members of the group; jurisdictional clarity; and leadership style.
- Personal factors, including: individual value systems; and personality characteristics.

Studies in organisation behaviour have shown that people are predisposed to tackle conflict in certain ways, according to their character rather than the context of the conflict. It is useful to identify these modes, and examine the utility of each. Thomas [1976] describes five orientations towards conflict handling (figure 3), based on areas of the joint outcome space:

Competitive - one participant seeks to dominate the process, without regard for the others. It is useful for quick decisive action, or where unpopular actions are perceived as necessary for important issues.

Collaborative - participants seek to understand their differences and achieve a mutually beneficial solution. It is appropriate where participants' insights and commitment are important and need to be merged rather than compromised.

Avoidant - the conflict is recognised to exist but is suppressed by one or more parties, or handled by withdrawal. It is useful where an issue is unimportant, where the potential disruption would outweigh the benefits of resolution, or where information gathering is more important.

Accommodative - a party becomes self-sacrificing to appease another, and places the other's interests above their own. It is useful when issues are far more important to one party than another, where one party is losing and needs to minimise loss, or simply to build harmony and gain social credits.

Sharing - each party makes some concessions in order to reach a compromise. It is appropriate where temporary settlements or expedient solutions are needed especially under time pressure, or where goals are directly opposed.

Each of these modes is appropriate in some circumstances; the more aware people are of the possibilities the more likely a suitable mode will be used. It is useful to compare these modes with the methods available for conflict resolution. For

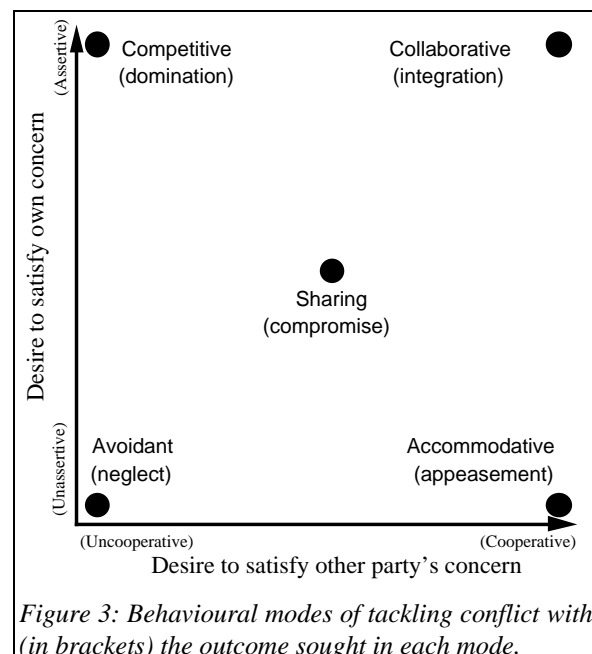


Figure 3: Behavioural modes of tackling conflict with (in brackets) the outcome sought in each mode.

example, collaborative methods such as negotiation and education, while most often used in the collaborative mode, can also be adopted in other modes. Education can be used to achieve conflict avoidance or accommodation by enabling participants to understand their differences better. Similarly, negotiation can assist with achieving a compromise, seeking an accommodation, or regulating competition. It is likely that successful negotiation requires at least some assertiveness and at least some co-operation from each participant. This in turn implies that each participant must have some motivation to resolve the conflict rather than avoid it.

A number of models have been proposed for conducting face-to-face negotiation in a commercial setting (e.g. Scott [1988], Fisher & Ury [1981], De Bono [1985]). Scott [1988] gives advice for preparation and the opening moments (setting the climate and procedure) of a negotiation. He uses a four stage model to pace the negotiation: Exploration; Bidding; Bargaining; and Settling. The exploration stage is emphasised as the most crucial, allowing participants to explore a range of possibilities before any confrontation takes place. In particular, it allows the participants to explain to each other their interests (figure 4), and discover shared goals which were previously obscured from both.

De Bono [1985] discusses the flaws in argumentation that render it ineffective as a means of negotiation. The adoption of a particular perspective (or theory) dictates how the world will be perceived and leads to a rejection of alternative theories, making argumentation a polarising process. The key to De Bono's method is the use of a third party to *design* solutions to conflict, as opposed to fighting, negotiating or problem solving. However, his complaints against negotiating and problem solving are based on very narrow definitions of these methods. Stefik *et al.* [1987] suggest that removing the personal attachment to positions prevents polarisation. Their computerised meeting room allowed participants to dispense with the feeling of ownership of ideas, and so reduce the associated emotions when ideas are discarded or adopted. Similarly, Fisher & Ury [1981] recommend that rather than bargaining over positions, participants should focus on interests, and investigate options for mutual gain.

Some of the recommendations made in these negotiation models are clearly useful for design. To summarise the main points:

- Exploration of each other's perspective is essential to constructive negotiation
- Participants should be separated from the bids in order to avoid polarization.
- The negotiation must begin by examining interests rather than positions so as to achieve a better understanding of the conflict.

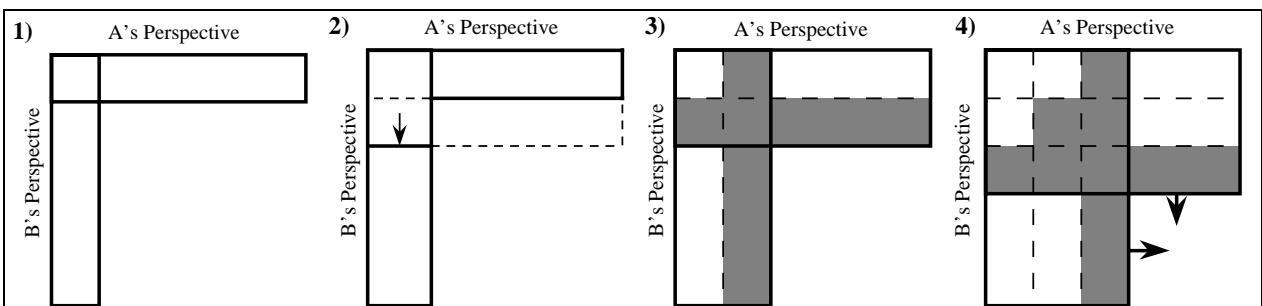


Figure 4: Scott [1988] explains the process of exploration in this way. In the first diagram, A and B's perspectives are very different, with little overlap. In (2), B begins to explain her perspective to A. Because A's perspective is different, A is able to take a great leap forward. In the process of learning more about the each others' perspectives, they discover joint interests, shown by the darker shaded areas. These areas are likely to yield a jointly acceptable solution to the conflict.

- There is a need to generate many options without discarding any prematurely.
- The procedure (and criteria) for selecting a good solution must be agreed on.

2.4. Conflict in Computing Science

Various fields of computer science have addressed the problems of conflict. The ubiquity of conflict in the real world means that attempts to model the world, most notably in AI, have to deal with it. Also, systems designed to support human interaction must take account of the conflict between people.

Knowledge-based systems rely on consistent knowledge for their inference mechanisms to work. In many cases, this is achieved by only consulting a single expert. Inconsistencies are attributed to mistakes in the acquisition process and are eliminated through a lengthy process of interactive debugging and refinement [Davis 1979]. This insistence that expertise must be consistent and rational imposes restrictions on the knowledge acquired. The knowledge acquisition process becomes not so much the modelling of the expert's behaviour, but the synthesis of a domain model which need not resemble any mental model used by the expert [Shaw & Woodward 1989]. In this way conflicts in the expert's knowledge can be filtered out. Although the same process of rationalisation can be undertaken with a group, the problems of conflict cannot be avoided so conveniently. There is little pressure on groups of experts to agree with one another, and the synthesis of a consistent domain model can be very difficult.

Shaw & Gaines [1988] compare the entity-attribute models of different experts, and identify four types of comparisons between conceptual systems:

- Consensus - experts use the same terminology to describe the same concepts
- Correspondence - experts use different terminology to describe the same concepts
- Conflict - experts use the same terminology to describe different concepts
- Contrast - experts use different terminology to describe different concepts

Each of these situations can be useful in capturing different perspectives, and in particular, the availability of alternative terminologies makes a knowledge-base more accessible.

Whilst Shaw's use of the term conflict is rather different from ours, it does highlight the way that terminological differences can obscure the deeper agreements and conflicts between people. Shaw describes a methodology for recognising each of the four situations using entity-attribute grids. However, this requires the participants to agree first on the definition of a common set of entities, and is concerned with differences in the way experts distinguish between the entities. It is not clear how this methodology might be extended to the types of representation used in systems analysis, where the establishment of an agreed set of entities is itself a problem of conflict resolution.

Distributed Artificial Intelligence (DAI) questions the usual AI assumption that a single self-consistent entity (such as a conventional knowledge base) can demonstrate intelligence. Problem-solving activities can be divided up among agents, according to their specialist knowledge, with the premise is that intelligence is an emergent feature of co-operative behaviour [Huhns 1987]. The paradigm has a natural ability to handle conflicting knowledge without the usual logical contortions arising from inconsistency, by allowing different agents to develop and maintain alternative hypotheses. This is demonstrated in the blackboard system [Nii 1986], in which separate knowledge sources communicate partial hypotheses using a shared *blackboard*. The knowledge sources modify and extend existing hypotheses on the blackboard.

Most DAI systems assume benevolent agents working towards the same goal. Rosenschein [1985] notes that in real world situations, perfect co-operation never happens, as the goals of any two agents will never coincide exactly. He uses payoff matrices from game theory to compare goals, and discusses various situations in which conflict of goals can occur, and how they can be resolved [Rosenschein & Genesereth 1985].

DAI has not progressed much beyond game theoretical studies of interaction [Stary 1991]. Models which require resource conflicts to be resolved involve little more than bidding and bargaining strategies. Keeping the contributions separate and applying them as separate reasoning systems avoids conflict, reducing it to a problem deciding which rule should be selected when several are applicable, where the possible combinations are foreseeable. Models which allow multiple hypotheses to be developed accept the first hypotheses to meet certain criteria, where the criteria used are built into the system. Work is needed to study how to combine elements of conflicting hypotheses as the basis for better hypotheses. In order to achieve this, the rationales behind each element of the hypotheses are needed.

Computer-Supported Co-operative Work (CSCW) studies how computers can be used in collaborative activities. Greenberg [1989] divides CSCW into two areas: Real-time collaboration and asynchronous collaboration. The former studies how computers are used to provide electronic media for use in face-to-face meetings, and to facilitate remote conferencing. Asynchronous systems co-ordinate group working over longer periods, and provide new communication channels between colleagues. These include electronic mail, bulletin boards and hypertext.

CSCW provides a number of tools intended to improve group collaboration and hence manage conflict. Xerox PARC's *CoLab* [Stefik et al 1987] is a testbed meeting room which uses high resolution screens to replace the role of the whiteboard. Two of the software tools provided are of interest here: *Cognoter* and *Argnoter*. *Cognoter* is used to develop outlines for talks and papers collaboratively, and divides meetings into three phases: *brainstorming*, *organising*, and *evaluation*. *Argnoter* is intended for use in evolving designs, and attempts to overcome three major causes of dispute: personal attachment to positions, unstated assumptions, and unstated criteria. Proposals are presented using webs of interconnected windows, often modifying existing proposals, or combining features of several. Reasons for and against are then linked to each proposal. Finally, the assumptions made by the arguments and criteria for decision making are elicited. The assumptions can be grouped together into *belief sets*, to characterise points of view, and to explore the consequences of those views. Stefik *et. al.* [1987] compare the tool to a spreadsheet, in that it does not understand the proposals and arguments that it manipulates, but can compute the logical relationships between them.

Hypertext offers the ability to support the collaborative elicitation and organisation of ideas over longer durations [Conklin 1987]. As Schuler [1988] points out, hypertext provides an excellent vehicle for supporting (but not supplanting) negotiation. Differing opinions and viewpoints can be represented and linked in the same system, encouraging plurality rather than stifling it. Lowe has built a hypertext system, SYNVIEW, based on a model for debate and reasoning [Lowe 1986]. By concentrating on debate, the system can contain all viewpoints rather than just a dominant one, with any group decision-making or voting based on access to a common body of material. Conklin [1989] takes this even further, and models positions, issues and arguments explicitly.

2.5. Conflict in Software Engineering

A number of recent studies have concluded that conflict has an important role to play in software engineering. For example, Curtis, Krasner & Iscoe [1988] identify *exceptional designers* who have a deep understanding of both the application domain and the design process which enables them “to integrate different, sometimes competing perspectives on the development process”. If the interaction of perspectives is to be supported, they need to be explicitly modelled. Each participant brings a number of preconceptions or biases into the software specification process, together with a certain amount of knowledge. The areas of knowledge do not fit together like a jigsaw, but instead overlap in some places, conflict in others, and often leave gaps. The process of elicitation and specification of requirements can be regarded as a knowledge acquisition and knowledge management task [Easterbrook 1989].

Finkelstein, Goedicke *et. al.* [1989] formalise the notion of a perspective into what they term a *ViewPoint*. Each viewpoint represents some area of knowledge and a preferred representation for that knowledge. More specifically, a viewpoint has the following components:

- a style, which is the representation scheme used;
- an area of concern, or domain;
- a specification, which is the set of statements in the viewpoint’s style describing the area of concern;
- a work plan, which describes how the specification can be changed, and any constraints on it;
- a work record, which describes how the specification developed, and its current status.

This model abstracts away from the people involved, allowing one person to have several viewpoints (as a person may have several areas of concern), and also for one viewpoint to represent several people (where people share an area of concern).

Finkelstein & Fuks [1989] use the viewpoint as a basis for a study of some of the communicational problems surrounding conflict. They describe a formal model of dialogue between two agents, which allows agents to share knowledge and detect inconsistencies between their knowledge. The dialogue consists of speech acts, such as statement, question, challenge, or withdrawal, and rules constrain which acts are legal in which contexts. Agents can query chains of reasoning made by other agents, and request information which they need to verify the conclusions. In this way, conflicts based on misunderstandings and incomplete knowledge can be detected and resolved.

Feather [1989] uses a basic specification as a point of departure for development along separate lines of concern. At some later stage these are merged to produce a single specification, which will then reflect all the concerns. This model has the benefit of delaying the resolution of conflict between separate concerns until after the information gathering stage. While it is not yet entirely clear how best to merge the parallel elaborations, Feather has examined the different types of conflict that occur.

One approach to easing the integration of separate specification components is through tools which support negotiation. Robinson [1990] describes tools that allow a single arbitrator to evaluate the preferences expressed by various perspectives, and guide the search for new solutions which satisfy all perspectives. A single domain model is used, expressed as a hierarchy of goals in which perspectives associate different values with the goals. Integration involves searching for novel combinations of proposals, which increase the satisfaction of all perspectives’ goals. This is done using a joint outcome space on which an ideal, but probably

unachievable combination of perspectives is used to stimulate consideration of other combinations that come close to this ideal.

These approaches to software specification all question the assumptions made by conventional software models which ignore conflict. Clearly, more work is needed to clarify how conflicts based on differing requirements can be resolved. One major issue is the need to establish common ground between viewpoints. No resolution can occur until participants have enough common ground to communicate; indeed, such common ground is needed before conflict can be expressed and recognised. In many of the above models, the common ground is assumed.

A second issue is how the resolution is devised. If all possible conflicts are foreseeable, then the resolutions can be enumerated beforehand, either directly, or using a set of heuristics. In general the design process is not predictable, and sociological studies of negotiation indicates that good resolutions require creative input. This poses problems for automatic tools. Anderson & Fickas [1989] suggest that in well charted domains, the experts will be aware of typical conflicts and how to deal with them, which suggests that resolutions can simply be elicited from experts. However, standard solutions are not necessarily the best, nor will they always work in the environment introduced by computerisation. The wealth of research on conflict in organisational behaviour suggests that at least in this field, resolutions are not well known.

2.6. Summary

We have discussed three broad areas in which conflict and conflict resolution are studied. The first of these, which we have labelled *mathematical models*, includes areas of decision theory and game theory. These fields are highly theoretical, and make some restrictive assumptions about, for example, the state of knowledge of the participants, and their motivations. Nevertheless, research in decision theory has developed a number of tools which can be used to clarify the issues in decision making. Game theory and bargaining have laid the groundwork to evaluating and understanding the use of strategies in conflict situations.

Behavioural models, from the social sciences, show more promise in channelling conflict situations into creative processes, but give little indication how such processes might be supported with computational tools. The main results from research in these areas is that not only is conflict inevitable in society, both within and between individuals and organisations, but that conflict has a useful role in facilitating change and producing higher quality group decisions.

Thirdly, we have covered areas within computer science which have studied conflict. Most of these fields are extremely young, some less than a decade old, and are only just recognising the role of conflict management. The fields we have covered seem to fall neatly into two camps: the AI approach, which attempts to automate completely the resolution of conflict between agents, based on mathematical approaches; and the supportive approach, in which computers are used as tools for supporting human conflict resolution, usually in a collaborative manner.

Finally, we have surveyed recent work in software engineering that recognises the existence of alternative and possibly conflicting views in requirements analysis. These models capture conflict by allowing different views to be represented, removing the conventional assumption that conflict can be avoided in software engineering. However, as yet no clear model exists of how conflict resolution might be modelled using these frameworks.

There is a consensus of opinion in work on group behaviour that good solutions to conflict require creative input. If this is the case, and it certainly seems likely in design contexts, then it is

unlikely that algorithmic conflict resolution methods can be provided. Rather, we can provide tools for analysing conflict situations and for encouraging co-operative and creative approaches to conflict resolution. The formal techniques developed in mathematical models of conflict can be used in these tools to supplement the creativity encouraged in behavioural models.

3. A MODEL FOR CONFLICT RESOLUTION

We now present a model for the management of conflict in the software design process. The twin goals of encouraging conflict and providing productive resolution methods form the basis for this model. In this paper we concentrate mainly on the latter goal, noting that the provision of explicit resolution methods and the associated communication channels should in itself begin to encourage the expression of conflict.

A support tool, *Synoptic*, has been implemented to demonstrate the feasibility of the model. *Synoptic* displays elicited descriptions side by side, allowing the participants to compare and annotate them. Discrepancies noted by the participants are then used by the system to prompt for underlying assumptions and issues. In effect, the system provides guidance and clerical support as the participants break the conflict down into a number of components in order to propose options for resolution.

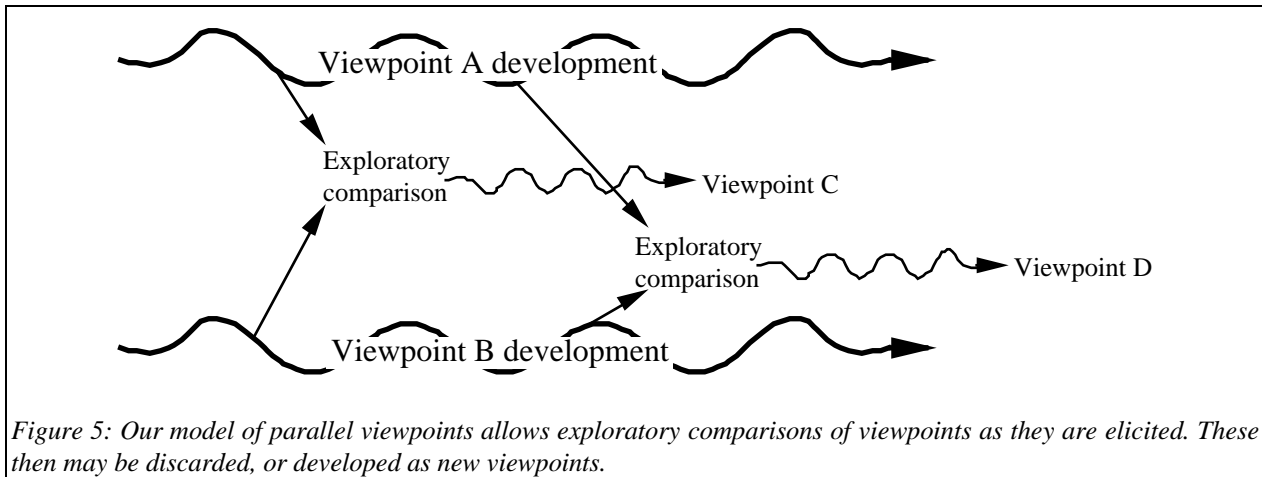
The model is intended to be prescriptive in that it acts as a set of guidelines, without being a rigid formal process. The tools developed to support the model are highly interactive, and are designed to provoke discussion of the conflict situations as much as elicit a suitable resolution. The model is based on the behavioural approaches used in organisational design, and in particular takes note of the need to separate the people from the problem, in order to avoid the polarising nature of arguing from entrenched positions. There are three phases in the model: an Exploration phase, a Generative phase, and an Evaluation phase. We discuss each of these phases in more detail in the rest of this section after a discussion of the context for this model.

3.1. Occurrence of Conflict in Software Design

As we noted in the introduction, avoidance seems to be the main reaction to conflict in existing software process models. There are several reasons why this ought not be the case. We have discussed the inevitability of conflict, and how its suppression makes resolution processes untraceable. Research into group behaviour indicates that conflict can produce higher quality solutions [Brown 1988]. Certainly, exploration of the areas where participants descriptions differ can lead to a much better understanding of the domain. This is a strong argument for conflict to be carefully managed in the software process, with participants encouraged to express divergent views. This will ensure that the resulting system does not reflect just one point of view, and does not ignore concerns which interfere with the dominant concern.

In software design, effective collaboration is essential. It is vital that there be no losers from any conflict in the specification process, as the commitment of all participants must be maintained [Easterbrook 1991]. Hence, encouragement of conflict must be matched with resolution methods which strive to satisfy all parties. An integrative approach should be adopted, to ensure that when divergent views arise they are incorporated into the process. The ultimate goal of the requirements process should be to produce a specification which represents all concerns.

The two key collaborative methods for conflict resolution are (integrative) negotiation and education. Both of these emphasise communication between participants, and both greatly ease conflicts based on communication problems. As Robbins [1974] notes, increased communication



leads to decreased conflict up to a certain level, but that too much communication can lead to increased conflict. A possible explanation is that a certain amount of communication allows participants to discover commonalities, iron out perceived conflicts, and correct misunderstandings. However, a high level of communication highlights the details on which participants do disagree. Such conflicts are likely to be well-founded, and should not be discouraged. However, arguing over trivial details can be counter-productive, and so a balance must be struck between encouraging communication and devoting appropriate amounts of effort to resolution of particular differences.

3.1.1. Specification Context

We use a notion similar to the viewpoints proposed by Finkelstein *et. al.* [1989] as a basis for our model of the specification process. Viewpoints allow the expression of conflict by providing alternative descriptions, while individually, each viewpoint remains consistent. Because viewpoints do not correspond to people, conflicts between a single person's roles can be represented, as well as inter-person conflicts. The term *viewpoint* refers to a formal object and we use the term *perspective* informally, to refer to the contribution made by a viewpoint. Each viewpoint has an *originator* – the person(s) associated with the viewpoint, and contains a self-consistent description of an area of knowledge.

The elicitation of a viewpoint does not occur in isolation from other viewpoints, as comparisons can shed new light on a particular perspective. Hence at any point in the process, parts of two or more viewpoints might be compared. The comparisons may involve conflict resolution, but are intended to be exploratory – any resolutions generated need not be incorporated in the final specification. Feedback from these explorations can be used in the development of the viewpoints, for instance to modify terminology, or to elicit information that the originator neglected. The results of any exploratory integrations can be treated as new viewpoints which can continue to take part in the development process (figure 5). Such derived viewpoints effectively represent coalitions of perspectives, which have been shown to arise in software projects [Curtis, Krasner & Iscoe 1988].

The modelling of viewpoints allows differences between perspectives to be captured and accommodated. If only a single description was maintained, differences between parties would tend to be avoided or suppressed, and often go unnoticed. As the viewpoints contain formal descriptions, it is possible to combine parts of different viewpoints to reason with, and detect inconsistencies. The process of parallel development of viewpoints – with exploratory

integrations being initiated at any point – provides the context for our model of conflict resolution.

3.1.2. Detection of Conflict

The first problem for conflict resolution is to recognise that a conflict exists. This might be harder than it seems for a number of reasons. The terminology used by the participants is unlikely to match exactly [Shaw & Gaines 1988], and the styles in which knowledge about an issue is expressed will differ. This difference may be because of different representation schemes, or different descriptions within the same representation scheme. Also, participants will have different areas and different amounts of knowledge, making it difficult to make comparisons. These problems make it hard to tell where participants are agreeing, let alone where they are disagreeing.

Our definition of conflict was based on interference: two parties are in conflict if the activities of one adversely affect the interest of another. Hence, viewpoints are free to differ, and only conflict when that difference matters for some reason, leading to interference. There are a number of situations in which the differences matter:

- When viewpoints need to be compared.
- When there is a need to reason with knowledge from several viewpoints.
- When the originators insist their viewpoints are “better” than others (and so perhaps should be adopted at the expense of them).
- When a coherent description is needed for further progress.

Under normal circumstances, differences between viewpoints are ignored, allowing them to develop independently. By only entering the conflict resolution process when differences between viewpoints matter, we avoid attempting to resolve conflicts unnecessarily. A conflict, then, is simply a difference that matters.

Note that defining conflicts as differences that matter will include many things that might not normally be regarded as conflicts. The distinction that Deutsch [1973] draws between real and apparent conflicts is deliberately ignored. Apparent conflicts here might include: where one party has misunderstood another’s position; where viewpoints use different terminology to describe the same thing; and where the interests at stake do not interfere, and can be combined directly. All these are treated as conflicts, and part of the task of the negotiation process is to identify what type of conflict has occurred, and hence whether or not a resolution is needed. The rationale for this approach is simple: it is impossible to tell without exploration whether a conflict is real or apparent, and this exploration is an essential step in the conflict resolution process anyway.

There is one remaining problem with which this approach does not help. Although the conflict resolution process is only invoked when differences become important, this does not ensure that all relevant viewpoints participate in the conflict resolution. When a difference between two viewpoints becomes important enough to attempt to resolve, there may be other viewpoints which contain extra information relevant for the resolution. It is not clear how these relevant viewpoints can be detected, especially in the presence of the mismatches in terminology and style discussed above. At present, the model assumes that all relevant viewpoints are included, and concentrates on the resolution.

3.1.3. Example

Take for example the specification of a library system. An analyst is trying to establish a state transition diagram for the books of the library, and is offered two different versions by two different librarians (figure 6). One gives a description based roughly on a book's physical whereabouts, whereas the other gives a description based on how a book can be accessed. There is a conflict between their views of the library.

While it may appear that there are a number of correspondences between the two descriptions, these are not as simple as they seem. For example, the concepts `ON SHELF` and `AVAILABLE` are similar, except that the latter includes books waiting to be shelved: it assumes that librarians be able to locate unshelved books for loan. `OUT` and `LENT` are also similar, except that the former includes books being used within the library, while the latter only includes such books if they are from the reserve collection. To make things worse, both could have used the same terminology.

3.2. Exploration phase

The first phase of the model is exploration. The aim of this phase is to arrive at a better understanding of the conflict. Essentially, this is a process of knowledge elicitation, as additional knowledge is needed about the descriptions in conflict. This phase involves identifying why the conflict occurred, and hence the type of conflict, the extent of the conflict, and the issues involved. Such information might be represented in a number of ways ranging from formal to informal, through a process of annotating the descriptions and linking elements of them together. In particular, links showing correspondences and discrepancies can be used.

The exploration phase begins once a conflict has been detected. The information available consists of the relevant parts of the viewpoint descriptions, and an indication of where the conflict was detected. For example, given the descriptions in figure 6, let us say that the analyst is trying to establish when a book is available for loan. The states `ON SHELF` in figure 6(a), and `AVAILABLE` in figure 6(b) seem to correspond roughly, but there is conflict, as neither the names, nor the transitions attached to these states match. In this case we begin the exploration with these two diagrams and an indication that the conflict is between `ON SHELF` and `AVAILABLE`.

The result of the exploration phase is a map of the conflict. This includes a list of correspondences and differences between the descriptions. In other words, the original disparity between viewpoint descriptions has been broken down into specific lists of items in the descriptions which correspond, and items which do not. Together, these comprise the *components* of the conflict. The exploration phase also elicits any issues underlying the

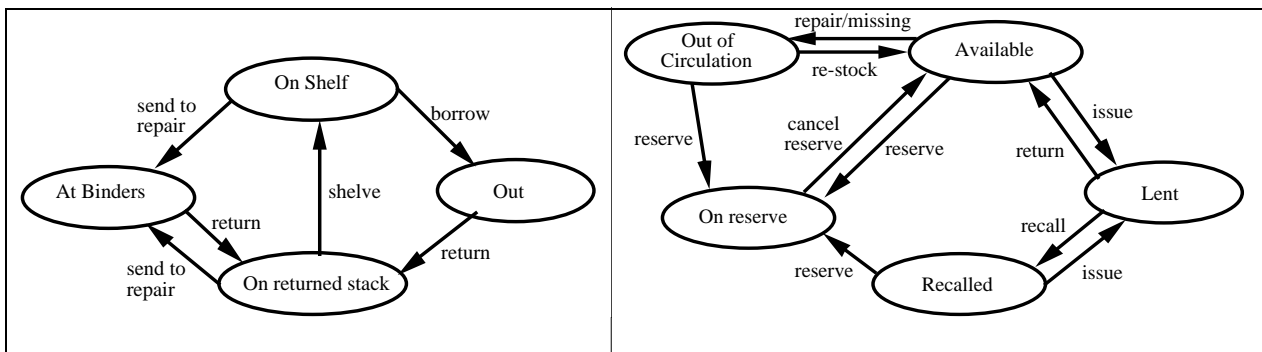


Figure 6: The possible states for a library book: (a) from the perspective of physical whereabouts; and (b): from the perspective of accessibility of a book.

correspondences and differences, and for each issue, criteria for its satisfaction.

3.2.1. Establishing Correspondences

The first problem is to establish some common ground between the descriptions. This is important to delimit the extent of the conflict, and to provide the participants with a basis for communication. The process starts with two descriptions, within which particular statements are known to conflict. To determine the extent of the conflict, the statements around the conflicting ones are compared, as these provide a context for the conflicting statements. Initially, this context consists of those statements in the original viewpoints which are directly connected to the ones in question. In a graphical notation, these are the arcs and nodes connected to the items in question, and for a chain of inference, all the immediate antecedents and consequences are used.

The problem is to identify correspondences between the items in the descriptions. Such correspondences may be exact or approximate. There is an exact correspondence if the items are agreed as having the same definition; the correspondence is only approximate if the meanings are similar, but differ in certain details. Many terminological differences will be discovered at this point: methodologies for recognising terminological mismatches, such as that of Shaw & Gaines [1988], as well as tools for detecting graph isomorphism, could be usefully employed here. The tool support provided by *Synoptic* is somewhat less sophisticated than these, relying on the user to identify correspondences.

To illustrate this process, consider the library books example. The arcs attached to the bubbles ON SHELF and AVAILABLE, can be compared. Firstly, we might note that there is a correspondence between SEND TO REPAIR and REPAIR/MISSING. The correspondence is not exact but it appears that the former is included in the latter. This raises the issue of how books going missing are handled in the first diagram, and whether this needs to be represented. The actions BORROW and ISSUE appear to be identical, but note that the return action in one diagram is the inverse of issue, while in the other, it leads to a new state, ON RETURNED STACK. In this case we can assume that the state AVAILABLE in the second diagram is the composition of ON RETURNED STACK and ON SHELF. Other correspondences can similarly be found, and items may be involved in more than one correspondence.

The examples described demonstrate different types of correspondence (figure 7). Most obviously there is equivalence, as in the case of ISSUE and BORROW (figure 7a). Often, different terms will be used to describe the same thing from different perspectives. In this case, one is the name of an action described by a librarian, and the other the same action described by a borrower. Both terms are useful, and could be recorded as synonyms. The comparison raises the issue of which should be used where.

As well as correspondences between single items, frequently groups of items will be linked. Where a single item in one description corresponds to a group in another, the representations are at different levels of decomposition (figure 7b). This is a common problem in systems analysis, as there is no standard way of deciding whether different parts of a description are at the same level of abstraction. The measure of level of abstraction is a subjective one, and different analysts will decompose the parts of a description in different orders. In many cases, such correspondences will not be exact, as decomposition usually reveals details about a description not considered at a coarser grain. Again, such comparisons yield issues that one description may not have addressed, which could be usefully discussed.

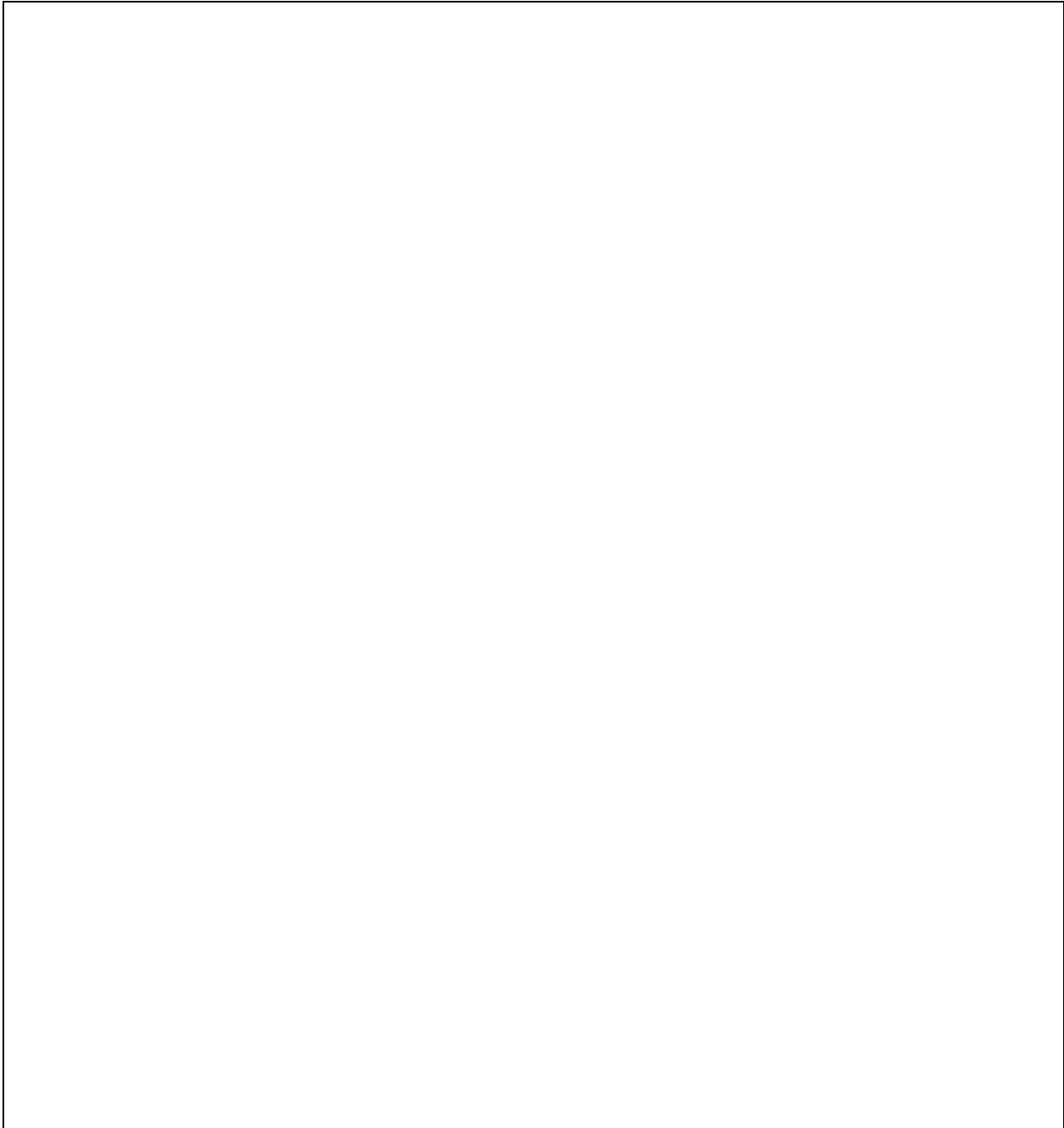


Figure 7: Correspondences between the descriptions of the library: (a) a correspondence between single items (although one of them recurs in the description); (b) a correspondence between a single item and a group; (c) a correspondence between two groups of items; and (d) an item for which there is no correspondence.

Correspondences between a group of items in one description and a different group of items in another description reveal where different types of decomposition have taken place. For example, the states `ON SHELF` and `ON RETURNED STACK` in the first description correspond to the group `AVAILABLE`, `ON RESERVE` and `RECALLED` in the second (figure 7c). In this example, both groups are decompositions of “In the Library”. The two groups will not necessarily match exactly. For example, the `RECALLED` state seems to include recalled books both before and after they are returned, and so is not totally captured within the group `ON SHELF / ON RETURNED STACK`. Different decompositions reveal different

concerns within the system modelling process, in the same way as Feather's parallel elaborations [Feather 1989].

Finally there is the case where an item or group of items in one description has no correspondence in the other. This may be because it has been omitted, or because the role played by such an item has been filled in other ways. For example, the *SHELVES* transition in the second description has no correspondence in the first (figure 7d).

The result of this stage is a list of correspondences between items in the viewpoints. Each correspondence may be recorded as exact or partial; but note that exact correspondences do not imply identical structure. The former indicate where there is agreement, and so restrict the area of conflict. However, where the correspondence is partial, there is still conflict to be resolved. In effect, the conflict has been broken down into its components: the initial rough description is replaced with a list of specific disparities between items in the descriptions.

3.2.2. Identifying the Conflict Issues

For a conflict to be resolved constructively, the reasons the parties are in conflict must be ascertained. These reasons may vary from lack of communication and poor understanding of other viewpoints, to differences in priorities and areas of concern [Robbins 1989]. Often the actual conflict is unrepresentative of the real issues which led to the conflict. Deutsch [1973] calls these *displaced conflicts*, and discusses the psychological reasons which cause people to express conflict in indirect ways. Although Deutsch's considerations are mainly to do with conflicts in personal relationships, there are other reasons why conflicts may appear displaced. For example, the descriptions being compared might be the result of long chains of development which are not necessarily based on the same initial assumptions and motivations.

Easterbrook [1989] notes that simply asking people to state any assumptions made by their descriptions is unlikely to be fruitful. There will be many assumptions, goals, and motivations involved in any description, some very trivial, and only a few will be relevant to the analyst. They are idiosyncratic in that what is obvious to one person may be an important decision to another [Kaplan 1989]. Discussion of assumptions must be prompted in some way, by asking "Do you assume X?" rather than "What do you assume?".

The systems gIBIS [Conklin 1989] and Argnoter [Stefik *et. al.* 1987] made use of the notion of *issues*, which are simply points that the design needs to address. They may take the form of suggested requirements (e.g. "The check-out process should ensure the borrower has not taken too many books"), or questions which need to be resolved (e.g. "How many books is too many?"). However, in these systems, issues are elicited unprompted: in gIBIS as a prelude to identifying positions and in Argnoter as supportive arguments for proposals. Our approach is to elicit issues only as a response to specific conflicts. Conflict provokes discussion of the issues, as participants raise any issues they feel other party's descriptions neglect. This prompting avoids having to ask participants simply to list any assumptions they made. It also avoids time wasted discussing issues on which there is already agreement, or which are irrelevant to the current context.

To assist with the elicitation of issues, four types of free-text annotation may be attached to the items in the descriptions, and to the correspondence links between items:

Comments - these are general purpose annotations, which can be attached to any item or group of items in the conflict. A typical use would be to attach to a correspondence between items to suggest a reason for the difference or similarity of items. Example: A comment might be

attached to the state ON RETURNED STACK noting “librarian B’s description does not include a returned stack”.

Assumptions - these are like comments but allow the user to note where a description appears to make some unstated assumption. These often arise when two descriptions are compared, and the comparison reveals issues that have been neglected in either description. Example: An assumption may be attached to the comment above, to note that “librarian B’s model assumes that books waiting to be shelved can be located for loan”.

Issues - these are points that need to be addressed. There are many circumstances under which issues arise, but often comments and assumptions will result in an issue. Example: the assumption above might lead to the issue: “How can books that have been returned but not shelved be traced?”.

Justifications - These are added to support a particular viewpoint or proposal. Often these will be added in response to assumptions and comments to provide a rationale for the original item. They will also be added in the next two phases of the process, to relate solution components to issues.

Several of the examples in the previous section showed how issues arise during the comparison of descriptions. Typically, issues are prompted by the creation of a correspondence, and the supporting tools prompt the user to note any assumptions or issues that arise when creating a correspondence. Assumptions have an issue attached automatically, questioning whether the assumption is reasonable, to ensure that the assumption is discussed when the issues are considered later in the process.

3.2.3. Agreeing Resolution Criteria

The final part of the exploration phase is the establishment of criteria by which to judge possible resolutions. Fisher & Ury [1981] suggest that objective criteria should be agreed before any resolutions are generated, to ensure that an agreement can be reached. This is to prevent participants moving the goal posts to get their personal preference accepted. However, it is often not clear before any solutions are proposed what the criteria should be. It is an open question as to how the prior agreement of criteria affects the generation of a resolution.

Our model treats the establishment of criteria as part of the exploratory phase, as it involves elicitation of further information from the participants. The criteria allow potential resolutions to be judged and compared. As the measures by which participants evaluate their satisfaction, they represent the participants’ goals for the resolution process. As such, these are the final part of the picture of the conflict built up by the exploration process. Issues represent the key points in the conflict; criteria show how the participants feel about these key points.

In our model, every issue has a related criterion describing desirable outcomes for that issue. In the simplest case, the issue asks a question, and the criteria attached to it provides an answer (often approximately), by reference to the participants’ concerns. Effectively we treat issues as points that the original viewpoints left unclear, and the criteria as the clarification of these points. In most cases, the participants will agree on the criteria for an issue with little difficulty, reserving disagreements for the order of priority among criteria. However, there are issues on which participants will define opposing criteria, and in this case, both are recorded, and the dispute added to the list of items in conflict.

Criteria can be used by participants to object to an issue. Issues are elicited in response to conflicts, and so will usually be agreed as being valid, if only because they are important enough to disagree over. However, occasionally, participants will object to an issue on the grounds that it

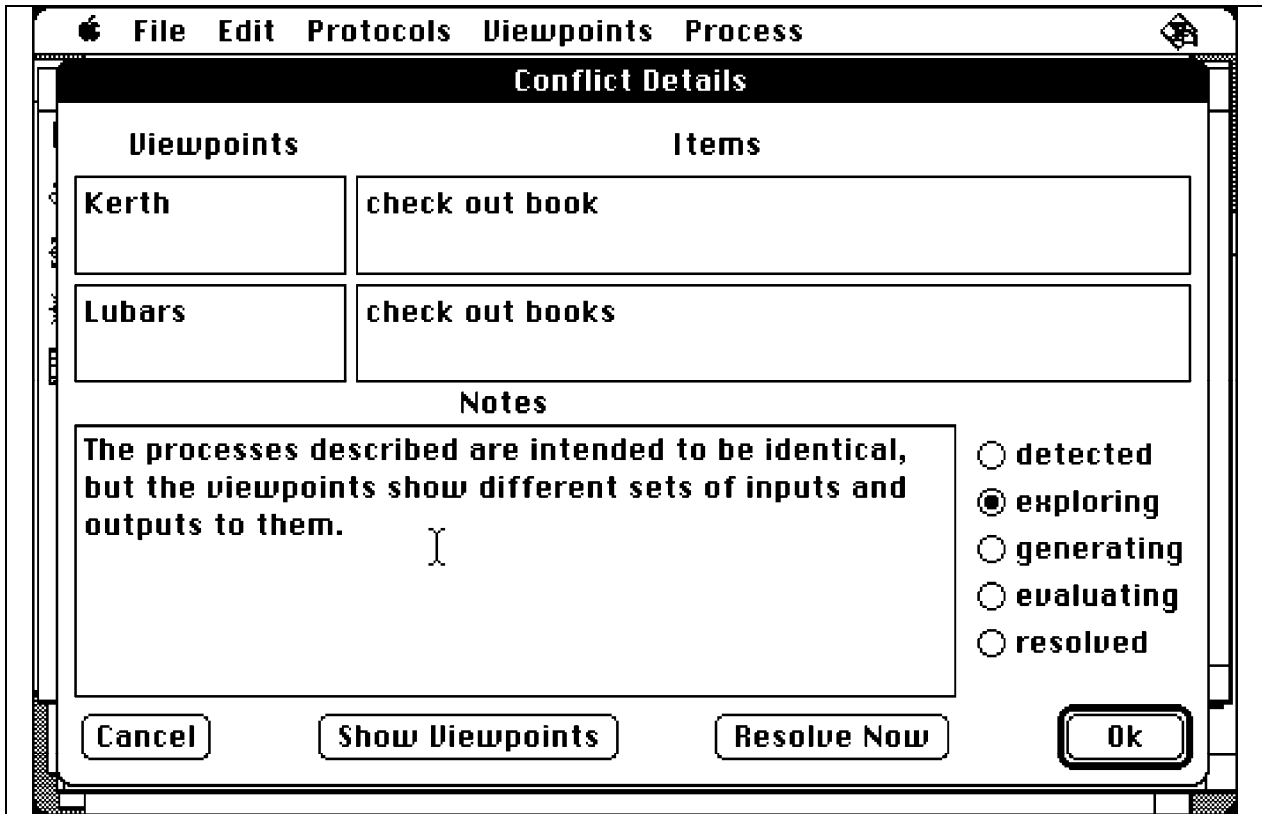


Figure 8: A screen snapshot from *Synoptic 1.0*, showing the form to be filled in when a conflict is detected. The system keeps track of the resolution process automatically.

is not really an issue, or is irrelevant. In this case, they can attach a null criteria, effectively stating “this issue can be ignored (in my opinion)”. Normally a null criteria will have either an assumption or a justification attached, explaining why. For example, a participant may object to the issue “There must be a way of locating unshelved books” because books can be assumed to be unavailable until shelved.

3.2.4. Functionality of the Exploration Tools

Synoptic is an extension of the *Analysier* system described in Easterbrook [1989]. It provides a set of tools to support the conflict resolution model. A single menu selects which phase of the model is in operation, and within each phase a palette of tools is available.

Conflicts between viewpoints can be noted by filling in a conflict form, as shown in figure 8. When a difference between viewpoints needs to be resolved, the conflict resolution process is invoked by selecting the exploration phase from the conflict menu. The same menu is used to move from one phase to the next, and to move back to a previous phase if necessary. The display of this menu is modified to show the current state: completed phases are marked with a tick, while phases beyond the next are shaded to show they are unavailable (see figure 9).

When noting a conflict, the user is asked to select those items in the viewpoint descriptions which are in conflict. In the exploration phase, these items, together with their immediate context are displayed side by side in a ‘synoptic’ window. A palette of tools is attached to this window to allow the following operations:

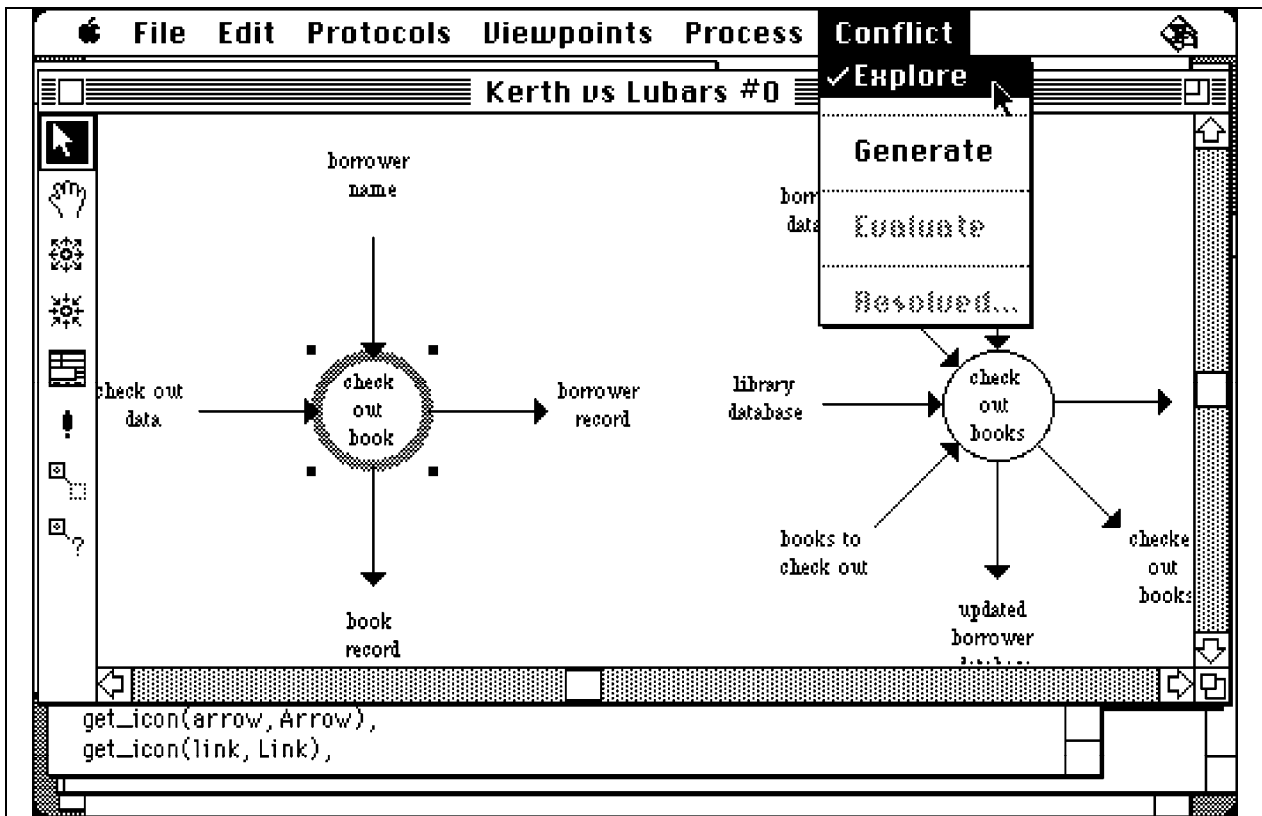


Figure 9: A screen snapshot from Synoptic 1.0, showing the window created to compare two descriptions, and the tool palate (down the left hand side). The “Conflicts” menu, which allows the user to move between phases is also shown.

- Selector (arrow icon) - for selecting items within the displayed descriptions, for some subsequent action, such as attaching a note. The selected items are displayed in grey.
- Mover (hand icon) - for moving a displayed description around. As items can be added or removed from the displayed descriptions, it may become necessary to adjust their relative positions within the synoptic window.
- Extend description (explode icon) - This tool extends descriptions in the synoptic window by adding more items from the source viewpoints. For any selected item in the synoptic window, all immediately connected items in the viewpoint description that are not already displayed in the synoptic window are added.
- Trim description (implode icon) - This tool allows the user to trim items from the descriptions displayed in the synoptic window. Items that are listed as part of the conflict on the conflict form cannot be trimmed.
- Conflict form (form icon) - This displays the conflict form.
- Attach note (exclamation mark icon) - This tool allows the user to attach a note to any item or link. The user will be asked to select the type of note to attach (see §6.3.2). Each type of note has a form to fill in. In the case of issues and assumptions, the form has slots for criteria and justifications. for any type of note the user is prompted for a brief title by which the note can be referred.
- Create correspondence (link icon) - A correspondence is created between the selected items. The user will be asked whether the correspondence is exact or approximate, and will be prompted for any issues to attach.

Find correspondence (link-question icon) - Displays any correspondences involving the selected items.

These tools allow the basic actions for the exploration stage. Note that *Synoptic* only plays a supporting role in this phase; the emphasis is on the human participants recognising and discussing correspondences. However, the tools perform clerical duties such as recording correspondences. Guidance is provided in that the system keeps track of task completion. For example, the system checks that all forms are filled in fully before allowing the user to move on to the next phase.

3.3. Generative Phase

The result of the exploration phase is a “map” of the conflict, which can be used to guide the search for possible resolutions. The second phase is concerned with generating these resolutions, and is essentially a design process. The aim is to propose solutions which overcome the limitations of the original viewpoints, and respond to the issues identified in the exploration phase. At this stage, the options are not evaluated, nor are they checked against the issues. This prevents the creative process being stifled by pragmatic considerations [Stefik *et. al.* 1987]. The options might be generated in a variety of ways, from directly combining elements of existing viewpoints to techniques such as lateral thinking and brainstorming.

The result of the generative phase is a list of options for resolution. These options are not intended to be complete resolutions, but might be combined in various ways to arrive at one. It is also possible that some of the options will be incompatible with one another: the evaluation phase will examine how the options can be combined.

3.3.1. Types of Conflict

Before the generative process gets underway, it is useful to characterise the type of each component of the conflict revealed by the exploration process. This will help to decide what form the generative phase will take, and what a possible resolution might consist of. We can identify three broad categories of conflict that might arise in systems analysis, as follows:

Conflicting interpretations - descriptions of the current situation or the current requirements do not match, usually because different perspectives interpret things differently. This category corresponds to the category *Beliefs* (or “how things are”), as described by Deutsch [1973].

Conflicting designs - suggestions (or partial designs) for how the system should be do not match. This roughly corresponds to the Deutsch’s category *Values*, or “How things should be”. While a requirements specification would not normally be expected to contain design information, participants are likely to express some of their requirements as partial designs, representing their preconceptions of the system.

Conflicting terminologies - the terms in which things are described do not match. This covers the communication problems suggested by Robbins [1989] as being a major cause of conflict.

In addition to these three categories of conflict, a scale for the severity of the conflict is used. This ranges from *non-interference* at one end to *mutual exclusion* at the other. The former implies the items in conflict can be combined directly without compromising either, whilst the latter indicates that each totally negates the other, and only one can be used (figure 10).

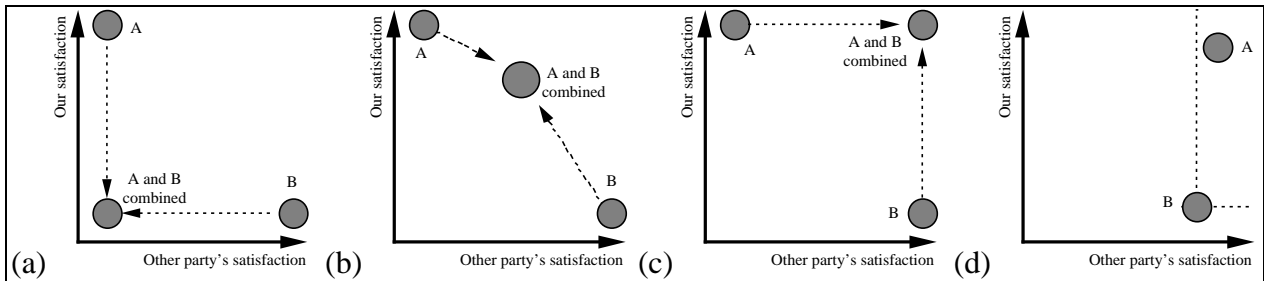


Figure 10: These diagrams show conflicts of different severity. In (a) the viewpoints are mutually exclusive, as their combination satisfies neither (the combination might not even be possible). In (b) the viewpoints can be combined, but with some loss of optimality for each party, and in (c) the viewpoints are non-interfering and can be directly combined. A variant of the non-interfering type is shown in (d), where one of the viewpoints already satisfies the other's concerns.

Using this schema, conflicts identified as non-interfering can be eliminated from further resolution work, as the direct combination of the two viewpoints provides an instant resolution. Where the two viewpoints provide alternative views or alternative terms, the circumstances under which each should be used still needs to be examined. For the remaining conflict types, there is plenty of scope for the design of novel resolutions which circumvent the conflict, by satisfying the underlying issues in other ways.

Table 1 describes typical examples of each of the categories and levels of severity, together with examples from the library books conflict. The examples are from the list of specific correspondences and differences discovered in the exploration phase. Some of the examples are phrased in a way that suggests possible resolutions; consideration of where these conflicts should appear in the table helped identify potential solutions. Note that these individual options are not exhaustive, and may obscure other possibilities. For example, exploration of this particular conflict revealed that one viewpoint was concerned with the physical whereabouts of a book, while the other is describing accessibility: it might make sense to retain both viewpoints entirely, and record for each book both its physical whereabouts and its loan status.

3.3.2. Generating Resolution Options

The model does not prescribe a particular method of generating resolutions. As already noted,

	Conflicting Interpretations	Conflicting Designs	Conflicting Terminology
non-interfering	Either interpretation can be used without affecting the other (need to find out which to use when). <i>Example: the possibility of books going missing has been omitted from the first viewpoint, and could be added directly if necessary.</i>	The design can be directly combined without compromising either. <i>Example: The recall facility, which is assumed to be a design suggestion, could be added directly to the first viewpoint</i>	Different terms have been used for the same concept (need to find out which to use when). <i>Example: "borrow" and "issue" apply to the same action. A borrower is more likely to use the former term, and a librarian the latter.</i>
partially interfering	Interpretations are not wholly consistent, and if both are to be used, some resolution is required. <i>Example: The "shelve" action is not wholly consistent with the second viewpoint as "available" does not quite correspond to "on shelf".</i>	Designs can be combined but interfere, and the direct combination may not be the ideal resolution. <i>Example: A reserve collection could be added to the first viewpoint by splitting the "on shelf" state to indicate the type of shelf.</i>	The same labels have been used for similar concepts. The differences need to be resolved. <i>Example: "Out of circulation" and "At binders". The latter is more specific, and implies that these books will eventually return.</i>
mutually exclusive	Interpretations totally contradict one another, and cannot be used in conjunction. <i>Example: There is no "return" action for recalled books in the second viewpoint, contradicting the notion of a returned book stack.</i>	Designs are completely incompatible, or tend to negate one another when combined.	The same labels have been used for different concepts, and some distinguishing terms are needed. <i>Example: The "return" from "at binders" is indistinguishable from the "return" from "lent". These might be completely different actions.</i>

Table 1: Different types and severities of conflict, and for each a description of the kind of situation covered, and an example from the library books conflict.

this is a design problem, and design methods are already well documented elsewhere (e.g. Finkelstein & Finkelstein [1983]). A range such methods might be usefully employed for generating options, depending on the components of the conflict, and the form of resolution required. Consideration of the category of the conflict components, as described in the previous section, may immediately provide one or more resolution options.

The categorisation of conflicts helps to determine what form a resolution should take. For example, conflicts in terminology, once detected, can be resolved fairly simply. Participants can be prompted for distinguishing terms if the same terms have been used for different concepts. Each such suggestion is a possible resolution, to be evaluated in the same way as other proposals. Where different terms have been used for the same concept, it is likely that both will be useful, and proposals will include circumstances under which a particular term might be favoured. In many cases, negotiating terminological differences is a waste of time, and participants should agree to differ, or make the effort to translate. It may be sufficient just for the participants to be aware of such conflicts.

Conflicting interpretations are slightly harder to resolve. Sometimes these will be based on incorrect information, which can be investigated. More often, they will arise from alternative ways of looking at things. Both interpretations might be useful, and proposals can be made which attempt to combine them, or which suggest circumstances under which one or other might be used. Proposals might also recommend that one interpretation should be discarded, in which case the issues raised by the discarded description need to be satisfied in other ways.

Conflicting designs involve a higher level of uncertainty. Often the conflict will be the result of conflicts not tackled at earlier stages, and the issues arising out of conflicting designs will indicate the concerns that lead to them. As the exploration stage has broken down the original conflict into its specific components, the designs can be examined more closely. Possible resolutions include combining the requirements underlying the designs, adapting one design to incorporate issues raised by another, or creating a totally new design which addresses the issues in new ways.

The result of this phase is a set of options for resolution. These will vary from the very specific (such as a particular change to a description), to entire viewpoints. Where an option is only applicable under certain conditions, these are also described as part of the option. Note that the original viewpoints could be considered as possible resolutions: one or other could be accepted unaltered, if it turns out to be a satisfactory resolution. In addition, some proposals will be candidates for combination to produce a more complete resolution, while others will be combinations which might need to be dismantled, if only a part is needed. At this stage, the proposals have not been evaluated or compared in any way.

3.3.3. Support for the Generative Phase

As the model does not prescribe any particular generative method, support for this phase is minimal. The system allows users to record resolution options as either free text, or replacement descriptions, or partial descriptions, annotated with justifications. Each recorded option has a space to record any conditions which might apply. Also, the user is prompted for a brief title by which the option can be referred.

Two tools are provided to assist in the process of generating these options. These represent the two main sources of resolution: consideration of the category and severity of each of the conflict components, and consideration of how each of the conflict issues might be satisfied. These tools

are relatively simple, in that they cycle through the conflict components, and the issues, encouraging the user to fill in the appropriate information, prompting for suggestions. Each suggestion is then recorded as a potential resolution. The tools do not enforce any particular ordering on this process, and the user may even move to the next phase without completing either of these tasks – in this case a warning message is displayed.

The support provided by *Synoptic* for the generative phase is relatively limited, and there are a number of ways in which this support could be improved. For example, information about the various ways in which particular types of conflict might be resolved needs to be encoded into *Synoptic*. This would allow the system to suggest possible resolutions and to ask leading questions to guide the user through the generation process.

3.4. Evaluation phase

The final phase, evaluation, consists of taking the options for resolutions and relating them both to the map of the conflict generated in the exploration phase, and to each other. The aim is to find the option or combination of options that best resolves the issues involved in the conflict. The approach is similar to that of the exploratory phase, and consists of linking items together and eliciting extra information to supplement the links.

The evaluation phase begins once a sufficient number of options has been generated. In fact, there is no distinct end to the generative phase. When participants feel that a good range of options has been generated, the evaluation phase can be initiated. The generative phase and the evaluative phase are kept deliberately separate, to prevent premature evaluation of the options from stifling generation of new suggestions. However, it is possible that the evaluation phase will also lead to new options, causing a cycling of these two phases.

3.4.1. Relating Options to Issues

The first task is to relate the suggested resolutions to the issues underlying the conflict. This may be done by taking each option in turn, and selecting the issues that it satisfies, or by taking each issue in turn, and deciding which options would satisfy it. Both approaches have merit, in that either may reveal additional links missed in the other. Satisfaction of issues is measured using the criteria attached to them.

The links between options and issues vary in the extent to which the option satisfies the criteria. Also, the relationship may be either positive or negative, where the former indicates the option contributes to the satisfaction of the issue, and the latter indicates it frustrates the issue. Unfortunately, the complex relationship between options and issues cannot be satisfactorily expressed using a simple numeric scale. Instead, a qualitative scale is used, along with explanatory notes. Participants may attach one of five values to each combination of option and issue. The values are: fully satisfies; partially satisfies; no effect; partially frustrates; and totally frustrates. The system attaches the value “no effect” by default. If the satisfaction of frustration is partial, an explanatory note is attached. These values will later be used to compare the options which contribute towards each issue.

3.4.2. Relating Options to One Another

The individual resolution options may interact in interesting ways. Some might usefully be combined to produce a resolution which satisfies more issues than either individually: for example, the suggestion of adding a “missing” state to the first viewpoint, and the suggestion of renaming the arrow from both this state and the “at binders” state to “restock” might be

combined to give a more complete solution. For other options, combination will negate some of the benefits: for example the suggestion of adding a reserve collection to the first viewpoint is not compatible with the suggestion of maintaining two types of state information, whereabouts and loan status. The range of interactions between options is analogous to the possible interactions between the parts of the original viewpoints, as shown in figure 10, which were evaluated using a scale of severity.

Where two or more options can be combined, the combination is recorded as a new option. In creating the combination, the way in which the combination satisfies the issues may need to be reconsidered. In most cases the combination will satisfy all the issues that the individual options satisfied. However, this is not always the case, and in particular, it is not clear how options with differing strength links with an issue might be combined. This information need to be elicited from the participants. Additionally, the combination might only be possible under certain circumstances, which need to be recorded as conditions for the new combined option.

3.4.3. Choosing a Resolution

Once the options have been linked to the issues and to each other, the only remaining problem is to select the best option or combination of options as a final resolution. In many cases, an agreed resolution will have emerged during the process, making much of the evaluation phase redundant. However, in cases where there is no obvious resolution, the options need to be compared. If there is an option (or combination) which satisfies all the issues, then this is a likely candidate. If any participants are unhappy with such a resolution, their reasons need to be elicited: these are likely to indicate issues that were missed in the exploration phase.

To a certain extent, if there is still no clear resolution at this stage, this can be seen as a failure of the negotiation process. The aim of the entire process is to explore the conflict and educate participants about each other's viewpoint: if this is successful, a resolution should emerge from the process, or the conflict should disappear. In the last resort, the participants might either agree some decision making procedure, or agree to leave the conflict unresolved. In the case of the former, the procedure will depend on the perceived importance of the conflict. An unimportant conflict might be resolved by an arbitrary method, while a more important conflict may require a process of ranking the issues, to select the option that best satisfies the most important issues.

The chosen resolution is represented as a new viewpoint which can be used instead of the original conflicting descriptions. Where the conflict involved only a part of the original viewpoints, the viewpoints can now inherit the relevant section from the resolution viewpoint. The original descriptions are retained as part of the record of the resolution process. The conflict map is recorded as a rationale for the resolution viewpoint, so that it is available for later re-examination if necessary.

3.4.4. Support for the Evaluation Phase

Linking issues to options is a straightforward interactive activity. Two approaches can be used: an option is displayed and the user asked to select those issues which it addresses, or an issue is displayed and the user asked to select the options which address it. The user can switch between these two approaches. In either case the procedure is the same: the option (or issue) is displayed alongside a list of the titles of the issues (or options) to which it may be linked. The full details for any title can be displayed by clicking on it. The user selects the relevant titles, and for each is prompted for the strength of the link, which is then indicated by flagging the title with one of the

symbols: ++, +, -, --, representing totally satisfies, partially satisfies, partially frustrates, and totally frustrates respectively. The links can have explanatory notes attached to them.

For the process of linking options to one another, the user may select a group of options to link together from a list, or may ask *Synoptic* to suggest a possible combination. In the latter case, combinations are chosen to maximise the number of issues satisfied, and may not always be sensible. The chosen combination is recorded as a new option, for which the process of linking to issues is repeated, as described above, with the links already filled in where possible. Where the link cannot be calculated automatically, for instance because one of the combination frustrates an issue which another satisfies, a question mark is displayed to remind the user to fill in the information.

Support for the final stage, selecting a resolution, is limited for the reasons set out in the previous section. An option is available to attach a numerical importance to each issue, so that the system can calculate a numerical score for each resolution option. The mechanics of this are very simple: the user is presented with each issue in turn and asked to select an importance value. These values are then combined with the values on the links between options and issues to generate a score for each option. The system does not allow for disagreement between participants over the importance measures. No attempt is made to support any other type of decision procedure.

4. SUMMARY

This paper has described a model of conflict resolution which can be used to integrate conflicting domain descriptions. This forms part of a larger model of requirements engineering based on the representation of multiple viewpoints, as described in Easterbrook [1991]. In recognition of the fact that carefully managed conflict can help eliminate errors and improve the quality of the requirements specification, the model encourages the expression of conflict by allowing participants to describe their viewpoints separately. Expression of conflict needs to be balanced with productive resolution methods, to encourage collaboration and to ensure that conflicts do not become counter-productive. The model described in this chapter was designed with this aim in mind.

The model consists of three phases: exploration of the participants' perspectives; the generation of suggestions for resolving the conflict, and the evaluation of these suggestions. During the exploration phase, the initial conflict is broken down into its components, represented as specific correspondences and differences between items in the viewpoint descriptions. These are annotated with comments describing any assumptions they make and issues they raise. These links and annotations act as a map of the conflict to guide the later stages. Resolution takes the form of designing novel ways of satisfying the issues. In the final phase, the ideas generated are then compared with one another and measured against the issues to determine the level of satisfaction. The option or combination of options which best satisfies the issues is chosen as a resolution.

The model combines the two most co-operative methods of conflict resolution: education and negotiation. Emphasis is placed on the exploratory phase in which participants learn about other viewpoints by comparing them to their own. Participants are encouraged to compare their viewpoint descriptions with others, and this comparison facilitates the elicitation of additional information, such as hidden assumptions. In fact, the final resolution is not necessarily the most important product of the negotiation process – the extra information elicited during the process, and the participants new understanding of one another's viewpoints may be far more valuable.

The entire process is highly interactive, and acts to structure the elicitation of additional information concerning the conflict. Rather than imposing a strict methodology on the participants, the various activities can be freely interleaved, so as to support discussion and exploration. Where an agreement is not reached, the arguments and understanding that have been built up aid judicial arbitration.

The model does have limitations. One of the weaknesses of the model is in the generative phase, where no guidance is offered regarding techniques for generating resolutions. More research is needed to determine which design techniques are appropriate for which types of conflict, so that guidance can be provided. Another weakness is that there is no way of ensuring that all relevant viewpoints participate in the conflict resolution. When a difference between particular viewpoints becomes important enough to attempt to resolve, there may be other viewpoints which contain extra information relevant for the resolution. It is not clear how these relevant viewpoints can be detected, especially in the presence of the mismatches in terminology and style discussed in section 1.1. If other viewpoints conflict with the generated resolution, then the resolution process may have to be repeated.

It is tempting to assume the model provides a general framework for conflict resolution. However, the model was designed to fill a specific need, that of comparing and merging previously elicited viewpoint descriptions. The analyst would carry out most of the work, usually over a period of time, involving the viewpoint originators when necessary. The model is not intended provide a form of meeting support, such as that provided in CoLab [Stefik et. al. 1987], nor is it intended to be a problem exploration tool, such as gIBIS [Conklin 1989], although it shares some features of these systems. It is possible, however, that the model could be adapted for use in these situations.

The support system, *Synoptic*, which was built to demonstrate the working of the model is a prototype. It demonstrates the basic processes of the model, but the support it provides is often minimal. Its main deficiency is that it requires the user to do much of the work. By incorporating more knowledge about the conflict resolution process, the system would provide much more guidance. For example, by comparing the issues which resolution options satisfy, the system might generate the most likely combinations of options, and use them to prompt the user for more information. Such a technique was used successfully in the knowledge acquisition system KSS0 [Shaw & Gaines 1987]. Another, major shortcoming of *Synoptic* is that it only allows two viewpoints to be compared at once.

Despite these shortcomings, *Synoptic* provides a solution to the problems of comparing viewpoints and resolving conflicts between them. We have demonstrated how example conflicts might be resolved using the system. Additional advantages include the ability to mix representations, and the elicitation of additional information. The former is important as it is notoriously difficult to compare knowledge represented in different ways. *Synoptic* provides a means to explore correspondences between different representations. In using the comparison of existing descriptions as a basis for exploration, the system is able to elicit underlying assumptions which might otherwise have remained hidden. Furthermore, the comparison process draws out the issues that the descriptions address.

5. ACKNOWLEDGEMENTS

This work was carried out at Imperial College, London. I am indebted to Anthony Finkelstein, for patient and perspicacious supervision, and to Bill Robinson, for useful discussions on the

ideas presented here. Much of the inspiration for this work came from a trip to the Knowledge Science Group at Calgary University, and special thanks are due to Brian Gaines for making this possible, and to all the other members of the group for providing food for thought. The work was funded by the SERC, studentship number 87311891.

6. REFERENCES

- Anderson, J. S., and Fickas, S. (1989) *A Proposed Perspective Shift: Viewing Specification Design as a Planning Problem*, Proceedings, Fifth IEEE International Workshop on Software Specification and Design, Pittsburg, Penn.
- Axelrod, R. (1984) *The Evolution of Co-operation*, Basic Books Inc, NY.
- Brown, R. (1988) *Group Processes: Dynamics within and between Groups*, Oxford: Basil Blackwell Ltd.
- Conklin, J. (1987) *A Survey of Hypertext*, Tech. Report No STP-356-86, Micro-electronics and Computer Technology Corporation (MCC), Austin, Texas. Reprinted in Greif, I., (ed) (1988) *Computer-Supported Co-operative Work: A Book of Readings*, Morgan Kaufmann, San Mateo, CA.
- Conklin, J. (1989) *Design Rationale and Maintainability*, Proceedings of the Twenty-Second Annual IEEE International Conference on System Sciences, Vol 2, Hawaii.
- Curtis, B., Krasner, H., and Iscoe, N. (1988) *A Field Study of the Software Design Process for Large Systems*, Communications of the ACM, 31 (11).
- Davis, R. (1979) *Interactive Transfer of Expertise: Acquisition of New Inference Rules*, Artificial Intelligence 12, p121-157.
- De Bono, E. (1985) *Conflicts: A Better Way to Resolve Them*, Penguin Books.
- Deutsch, M. (1973) *The Resolution of Conflict*, Yale University Press, New Haven.
- Easterbrook, S. M. (1989) *Distributed Knowledge Acquisition as a Model for Requirements Elicitation*, Proceedings, Third European Workshop on Knowledge Acquisition for Knowledge Based Systems (EKAW-89), Paris.
- Easterbrook, S. M. (1991) *Elicitation of Requirements from Multiple Perspectives*, PhD Thesis, Imperial College, University of London.
- Feather, M. S. (1989) *Constructing Specifications by Combining Parallel Elaborations*, IEEE Transactions on Software Engineering, 15 (2), Feb 1989, p198-208.
- Finkelstein, A. C. W., Goedicke, M., Kramer, J., and Niskier, C. (1989) *ViewPoint Oriented Software Development: Methods and Viewpoints in Requirements Engineering*, Proceedings, Second Meteor Workshop on Methods for Formal Specification, Springer-Verlag, LNCS..
- Finkelstein, A. C. W., Finkelstein, L. and Maibaum, T. S. E. (1990) *Engineering-In-The-Large: Software Engineering and Instrumentation*, Proceedings, UK IT '90, pp 1-8, Peter Peregrinus.
- Finkelstein, A. C. W., and Fuks H. (1989) *Multi-Party Specification*, Proceedings, Fifth IEEE International Workshop on Software Specification and Design, pp185-195.
- Finkelstein, L., and Finkelstein, A. C. W. (1983) *Review of Design Methodology*, IEE Proceedings, 130 (4), June 1983.
- Fisher, R., and Ury, W. (1981) *Getting to Yes: Negotiating Agreement Without Giving in*, Hutchinson.
- Greenberg, S. (1989) *A Survey of Computer-Supported Co-operative Work*, Draft Report, Alberta Research Council, Calgary, Canada.
- Huhns, M. N. (ed) (1987) *Distributed Artificial Intelligence*, Morgan Kaufmann Publishers Inc, Los Altos CA..
- Kaplan, S. M. (1989) *COED: Conversation-Oriented Software Environments*, Draft Report, University of Illinois at Urbana-Champaign.
- Keeney, R. L., and Raiffa, H. (1976) *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*, J. Wiley & Sons, NY.
- Lehman, M. M. (1990) *Uncertainty in Computer Application*, Technical Correspondence, Communications of the ACM, 33 (5), May 1990.
- Lowe, D. G. (1986) *SYNVIEW: The design of a system for co-operative structuring of information*, Proceedings, Conference on Computer-Supported Co-operative Work, Austin, Texas, p376-385.
- Luce, D. L., and Raiffa, H. (1957) *Games and Decisions: Introduction and Critical Survey*, J. Wiley & Sons, NY.
- Nii, H. P. (1986) *Blackboard Systems (part 1): The Blackboard Model of Problem Solving and the Evolution of Blackboard Architectures*, The AI Magazine, Summer 1986.
- Patchen, M. (1970) *Models of Co-operation and Conflict: A Critical Review*, Journal of Conflict Resolution, 14, (3), Sept 1970.
- Rapoport, A., (ed) (1974) *Game Theory as a Theory of Conflict Resolution*, D. Reidel Publ. Co., Dordrecht, Holland.

- Robbins, S. P. (1974) *Managing Organizational Conflict: A Non-traditional Approach*, Prentice Hall, NJ.
- Robbins, S. P. (1989) *Organizational Behaviour: Concepts, Controversies, and Applications*, (fourth edition) Prentice Hall, NJ.
- Robinson, W. N. (1990) *Negotiation Behaviour During Multiple Agent Specification: A Need for Automated Conflict Resolution*, To appear, ICSE-90.
- Rosenschein, J. S. (1985) *Rational Interaction: Co-operation Among Intelligent Agents*, Ph.d. Thesis, Report No STAN-CS-85-1081, Dept of Computer Science, Stanford University, Stanford, CA..
- Rosenschein, J. S., and Genesereth, M. R. (1985) *Deals Among Rational Agents*, Proceedings, Ninth International Joint Conference on Artificial Intelligence, p91-99.
- Schuler, D. (1988) *AI and Hypertext in Support of Negotiation*, in Bernstein, M., (ed) (1988) Proceedings, AAAI-88 Workshop on AI and Hypertext: Issues and Directions .
- Scott, B. (1988) *Negotiating: Constructive and Competitive Negotiations*, Paradigm Publishing, London.
- Shaw, M. L. G., and Gaines, B. R. (1988) *A Methodology for Recognising Consensus, Correspondence, Conflict, and Contrast in a Knowledge Acquisition System*, Proceedings, Third Knowledge Acquisition For Knowledge-Based Systems Workshop, Banff, November 1988.
- Shaw, M. L. G., and Woodward, J. B. (1989) *Mental Models in the Knowledge Acquisition Process*, Proceedings, Fourth Knowledge Acquisition For Knowledge-Based Systems Workshop, Banff, October 1989.
- Stefik, M., Foster, G., Bobrow, D. G., Kahn, K., Lanning, S., and Suchman, L. (1987) *Beyond the Chalkboard: Computer Support for Collaboration and Problem Solving in Meetings*, Communications of the ACM 30 (1).
- Strauss, A. (1978) *Negotiations: Varieties, Contexts, Processes and Social Order*, Jossey-Bass Publishers, San Francisco, CA.
- Thomas, K. (1976) *Conflict and Conflict Management*, in Dunnette (ed), Handbook of Industrial and Organizational Psychology, Rand McNally College Publ. Co.