Viewpoints: principles, problems and a practical approach to requirements engineering

Ian Sommerville and Pete Sawyer

Computing Department, Lancaster University, Lancaster LA1 4YR, UK

The paper includes a survey and discussion of viewpoint-oriented approaches to requirements engineering and a presentation of new work in this area which has been designed with practical application in mind. We describe the benefits of viewpoint-oriented requirements engineering and describe the strengths and weaknesses of a number of viewpoint-oriented methods. We discuss the practical problems of introducing viewpoint-oriented requirements engineering into industrial software engineering practice and why these have prevented the widespread use of existing approaches.

We then introduce a new model of viewpoints called *Preview*. Preview viewpoints are flexible, generic entities which can be used in different ways and in different application domains. We describe the novel characteristics of the Preview viewpoints model and the associated processes of requirements discovery, analysis and negotiation. Finally, we discuss how well this approach addresses some outstanding problems in requirements engineering (RE) and the practical industrial problems of introducing new requirements engineering methods.

1. Introduction

The specification of large computer-based systems is a very complex process. Ideally, a system specification should only define what services the system should provide and the operational constraints on the system. It should be complete and consistent, should not include implementation detail and should be presented in such a way that it is understandable by a range of readers from potential end-users of the system to engineers responsible for its implementation.

In practice, these are unrealisable goals. For technical, human and environmental reasons, system requirements specifications will always be imperfect. However, although perfection is impossible, there is no doubt that much can be done to improve the quality of most system specifications. It has been recognised for many years that problems with specifications are probably the principal reason for project failure where systems are delivered late, do not meet the real needs of their users, and perform in an unsatisfactory way [GAO 1979; Gibbs 1994; Barlas 1996].

Improving the quality of specifications can be achieved in two ways:

1. By improving the requirements engineering process so that errors are not introduced into the specification.

2. By improving the organisation and presentation of the specification itself so that it is more amenable to validation.

This paper discusses an approach to system requirements engineering which addresses both of these improvement dimensions. This is based on collecting and analysing the requirements for a computer-based system from different perspectives or viewpoints. Viewpoints are entities which may be used to structure the process of requirements elicitation and to structure the requirements specification.

In the remainder of the paper, we introduce the notion of viewpoints, describe several viewpoint models which have been proposed and practical problems of introducing these into industrial software engineering. We go on to propose a flexible, 'lightweight' model of viewpoints (Preview) which has been designed to be incorporated into existing requirements engineering processes at relatively low-cost. Finally, we assess the utility of this approach.

2. Viewpoints

A viewpoint-based approach to requirements engineering recognises that all information about the system requirements cannot be discovered by considering the system from a single perspective. Rather, we need to collect and organise requirements from a number of different *viewpoints*. A viewpoint is an encapsulation of partial information about a system's requirements. Information from different viewpoints must be integrated to form the final system specification.

The principal arguments in favour of a viewpoint-based approach to requirements engineering are:

1. Systems usage is heterogeneous – there is no such thing as a typical user. Viewpoints may organise system requirements from different classes of system end-user and other system stakeholders.
2. Different types of information are needed to specify systems including information about the application domain, information about the system's environment and engineering information about the system's development. Viewpoints may be used to collect and classify this information.
3. Viewpoints may be used as a means of structuring the process of requirements elicitation.
4. Viewpoints may be used to encapsulate different models of the system each of which provides some specification information.
5. Viewpoints may be used to structure the requirements description and expose conflicts between different requirements.

The notion of viewpoints for requirements engineering is intended to formalise what we believe is a natural part of any analysis process. A good requirements engineer will collect information about whatever is being studied from a number of different sources and will recognise that these sources may have different, often equally valid, perspectives. Recognising these perspectives and reconciling differences between them is essential if the analysis is to be valid. Viewpoint-oriented approaches are simply a means of formalising this intuitive multi-perspective analysis.

In the viewpoint-based methods which have been developed, two different kinds of viewpoint have been proposed:

1. *Viewpoints associated with system stakeholders.* Informally, a system stakeholder is anyone who is, directly or indirectly, affected by the existence of a system. Hence stakeholders may be end-users of a system, managers of organisations where systems are installed, other human and computer-based systems in an organisation, external entities who have some kind of interest in the system (e.g., regulatory bodies, customers of an organisation which has installed the system) and engineers involved in the design, development and maintenance of the system.
2. *Viewpoints associated with organisational and domain knowledge.* Organisational and domain knowledge is knowledge which constrains the system requirements. The constraints may be physical (e.g., network performance), organisational (e.g., incompatible hardware used in different divisions of a company), human (e.g., average operator error rate) or may reflect local, national or international laws, regulations and standards. This type of viewpoint cannot be associated with a single class of stakeholder but includes information collected from many different sources (people, documents, other systems, etc.).

To illustrate the range of viewpoints which may be useful for discovering, analysing and documenting the requirements for a computer-based system, consider a (relatively simple) Internet-based system to provide on-line banking facilities to customers through a WWW interface. Viewpoints which may have to be considered are:

1. One or more customer viewpoints, depending on different types of customer such as business and personal customers (stakeholder).
2. One or more bank staff viewpoints covering the operation and management of the system (stakeholder).
3. A security viewpoint (stakeholder).
4. A marketing viewpoint (stakeholder).
5. A database viewpoint (organisational).
6. A personnel viewpoint (organisational).
7. A regulatory viewpoint (domain–banks have external regulators).
8. An engineering/networking viewpoint (domain).

In practice, if too many viewpoints are identified, it is difficult to manage the large amount of information generated and prioritise the requirements. Therefore, an essential stage of most viewpoint-based methods is to select only the most critical viewpoints to be used in the analysis. Requirements engineers must balance the advantages of wider coverage offered by additional viewpoints against the costs of analysis and the problems of information management.

This paper focuses on viewpoints for requirements engineering of computer-based systems but the concept has also been found to be useful in other areas. In intelligent teaching systems and distributed AI, the notions of cognitive viewpoints to encapsulate beliefs have been found to be useful [Moyse 1992; Self 1992]; in communications, the ODP reference model defines five viewpoints (enterprise, information, computational, engineering and technology) from which a system may be specified [Linington 1995; Bowman *et al.* 1996]; in CSCW, viewpoints have been used to structure organisational analyses [Hughes *et al.* 1995] and in concurrent engineering, viewpoints have been proposed as a systematic approach to conflict analysis [Klein 1992].

To avoid an information explosion, we have restricted the discussion here to approaches which have adopted the explicit notion of a viewpoint rather than a more general multiple perspective approach to analysis. This more general notion is supported in the approach to requirements engineering proposed in the NATURE project [Jarke *et al.* 1993; Zemanek *et al.* 1995] by Jackson and Jackson [1996] and by Feather [1993].

3. Viewpoint models for requirements engineering

A number of different models of viewpoints have been developed as part of different multi-perspective approaches to requirements engineering. Different models are applicable at different stages in the requirements engineering process. Of course, we recognise that there is no such thing as a 'standard' requirements engineering process but we believe that most processes broadly conform to the general model illustrated in Fig. 1.

The cloud icons in this model indicate that the boundaries of these activities are not well-defined. The inputs to the model are a statement of organisational needs plus information on existing systems which impact the system to be procured in some way. These may be directly interfaced to the system or may have to be reused, in some way, as part of the system development. Other inputs include applicable standards and regulations, and domain experience.

The phases of this generic requirements engineering process are:

1. *Requirements discovery.* Given a statement of organisational needs, various different sources are consulted to understand the problem and the application domain and to establish their requirements. These requirements may not be complete and may be expressed in a vague and unstructured way

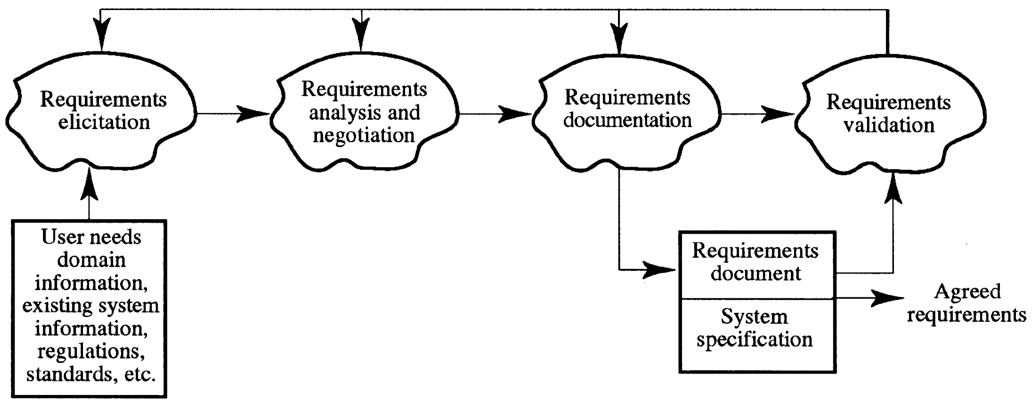

Requirements elicitation
Requirements analysis and negotiation
Requirements documentation
Requirements validation
User needs domain information, existing system information, regulations, standards, etc.
Requirements document
System specification
Agreed requirements

Figure 1. The requirements engineering process.

2. *Requirements analysis.* The requirements collected during the discovery phase are integrated and analysed. Usually, this results in the identification of missing requirements, inconsistencies and requirements conflicts. The discovery phase generally has to be re-entered to find additional information to resolve these problems.
3. *Requirements negotiation.* The system stakeholders negotiate to agree on a set of requirements for the system. Generally, there are a greater number of desirable requirements than can be implemented so decisions have to be made at this stage about leaving out requirements and modifying requirements to result in a lower-cost system.
4. *Requirements specification.* The set of agreed requirements is documented. The output from this process may be either a user requirements document, a system specification or both of these. A user requirements document is usually a natural language document where the system requirements are set out in a form understandable by customers and end-users of the system. A system specification is a more detailed description of what services the system should provide and the constraints on its development and operation.

This process model is consistent with Freeman and Leite's discussion [Leite and Freeman 1991] of requirements engineering activities. They suggest that these activities fall into two classes:

- Elicitation activities which are concerned with fact finding, communication and fact validation.
- Modelling activities which are concerned with requirements representation and organisation.

In our model, the elicitation activities are requirements discovery, analysis and negotiation; requirements specification is a modelling activity.

3.1. Viewpoints for requirements elicitation

Where viewpoints are used to support requirements elicitation, they are clearly outside of the system being specified and are, primarily, the sources of the system requirements. At this stage, only informal descriptions and models of the system are likely to be available. Requirements are usually expressed using natural language, diagrams and domain-specific notations such as mathematical formulae, etc.

Leite and Freeman [1991] describe an approach to viewpoint-oriented elicitation where they consider a viewpoint to be:

> a standing or mental position used by an individual when examining or observing a universe of discourse. It is identified by an individual, e.g., his name, and his role in the universe of discourse, e.g., a systems analyst, programmer or manager.

From this definition, they go on to describe an approach to requirements elicitation based on collecting information from different viewpoints and constructing *views*. A view is an integration of different types of information (e.g., data processing information, data structure information) taken from the same viewpoint.

An interesting characteristic of this work is that there is a one-to-many mapping from information sources to viewpoints. The viewpoint is a mental position so the same individual may provide information from several different viewpoints. This reflects the fact that people often have a number of different roles in an organisation (e.g., the same person can be a system end-user and responsible for overall system security). The strength of this work is that it provides for automatic checking and problem detection at an early stage of the elicitation process.

We have also been involved in previous work on viewpoints for requirements elicitation. In the first version of this work [Kotonya and Sommerville 1992], we modelled a viewpoint as a receiver of services from a system and a provider of data and control information. Viewpoints were therefore clearly associated with entities which interacted with the system to be specified. These interactors could either be system end-users or other computer systems which were interfaced to the system being specified.

The arguments in favour of this approach are:

1. It is relatively easy to decide whether or not an entity is a valid viewpoint. If it interacts with the system in some way, it is a viewpoint and the interaction must be defined.

2. A large class of systems may be modelled as service delivery systems. This is confirmed by Greenspan and Feblowitz [1993] who discuss the advantages of this approach. It is particularly appropriate for interactive systems. End-users find it fairly natural to describe their requirements in terms of services as, for example, they can relate services to the support of their own business processes.

3. Non-functional requirements such as performance and availability requirements can be associated with the delivery of a service or a set of services. They are

expressed as service constraints [Kotonya and Sommerville 1993]. These may differ from one viewpoint to another and part of the viewpoint analysis process is to detect and resolve such conflicts.

4. Control information associated with the starting, stopping and delivery of the services may be naturally included with the service description.
5. Specific user interface requirements may be associated with services delivered through the interface.

A method called VORD (Viewpoint-Oriented Requirements Definition) and associated tools were designed to support this model of viewpoints. The VORD process included activities concerned with viewpoint identification, viewpoint service description, cross-viewpoint analysis to discover inconsistencies, omissions and conflicts and developing an object-oriented model of the system from the viewpoint analysis. VORD allowed the services to be specified in any appropriate notation either informal, structured or formal. Multiple specifications, in different notations, of the same services could be provided and linked.

An evolution of the VORD method [Kotonya and Sommerville 1996] introduced the idea of indirect viewpoints. Indirect viewpoints may express requirements on a specific service, on a set of related services or on all of the services provided by the system. This is a broad notion which encompasses viewpoints such as engineering viewpoints which are concerned with the system design and implementation, organisational viewpoints which are concerned with organisational issues such as process re-engineering and external viewpoints such as the viewpoint of a regulator responsible for certifying safety-critical systems.

3.2. Viewpoints for requirements modelling

As part of the detailed requirements specification of a system, it is common practice to develop a set of system models which may be expressed in informal, structured or (less commonly) mathematically formal notations. There is an important strand of work in viewpoint-oriented requirements engineering which considers a viewpoint to be a framework for identifying, defining and checking the consistency of these system models.

Structured methods, such as JSD [Jackson 1983] or the various flavours of object-oriented analysis [Rumbaugh *et al.* 1991; Jacobson *et al.* 1993; Booch 1994], incorporate an implicit notion of viewpoints. These methods suggest that a number of different system models should be developed, each of which can be thought of as a different viewpoint on the system. SADT [Ross 1977; Schoman and Ross 1977] goes further in that it suggests that analysis should be undertaken from different viewpoints but it does not include viewpoints as explicit entities in the method. Rather, viewpoints are seen as sources or sinks for data in a data-processing model of the system which is required by the method.

The CORE method [Mullery 1979] which is based on SADT-like 'data processing' models of a system was the first structured method to include viewpoints as method entities. It includes explicit viewpoint identification and analysis steps as part of the method. CORE has been widely used for UK defence projects and in the European aerospace industry but has made less of an impact outside these domains. There is no good public documentation on the method or low-cost, readily available supporting tools.

In structured methods, the set of models, the notations used, and the rules and guidelines applied to these models are pre-defined. The system models can only be presented from the perspectives defined in the structured method. This can lead to misunderstandings because of the misfit between the customer and end-users perception of the system requirements and the system description. The method rules are often inappropriate as requirements are evolving so that the method-support tools are a hindrance rather than a help.

A complementary approach to system modelling with viewpoints is to define the viewpoints from which models should be developed without constraining the model representation. The models developed are not necessarily limited to models of the software whose requirements are being specified but may include organisational models, process models, etc. For example, Greenspan and Feblowitz [1993] define four viewpoints which may be used in deriving system requirements:

1. A service viewpoint where a set of services provided to customers is modelled. These services may be provided through an automated system, by people or by some mixture of the two.
2. A set of workflows or work processes which provide the services.
3. A model of the organisation responsible for service provision.
4. A model of the set of systems which provide the capabilities and the resources for providing the services.

The approach of pre-defining a set of viewpoints and constructing models based on these viewpoints has the merit of simplicity. It also limits the amount of information which has to be collected and provides a basis for the requirements discovery and elicitation process. However, it is restrictive in that the models proposed may be inapplicable in some domains and inappropriate for some types of system. For example, we agree with Greenspan that the service-oriented perspective is a useful one for many types of system. However, we found that it was artificial (and difficult) to apply this approach to real-time control systems.

The approach proposed by Nuseibeh *et al.* [1994] is a flexible approach which allows specification from multiple viewpoints without pre-defining the notations which should be used. Viewpoints are defined as "loosely coupled, locally managed, distributable objects which encapsulate partial knowledge about a system and its domain, specified in a particular, suitable representation scheme, and partial knowledge of the process of development." They are structured encapsulations of information with five slots:

1. A representation style which defines the notation used in the specification.
2. A domain which is defined as 'the area of concern addressed by the viewpoint'.
3. A specification which is a model of a system expressed in the defined style.
4. A work plan, with a process model, which defines how to build and check the specification.
5. A work record which is a trace of the actions taken in building, checking and modifying the specification.

This approach to viewpoints can be considered as a 'method' in its own right but it is perhaps more appropriate to view it as a meta-method which is used to define requirements engineering methods for use in a specific application domain or organisation. A method engineer must define a set of viewpoint templates (including the viewpoint consistency rules) which are then instantiated during the requirements engineering process.

A novel feature of this approach is its notion of inconsistency management [Easterbrook and Nuseibeh 1996]. The system does not enforce consistency across viewpoints but provides support for detecting and, if necessary, resolving inconsistencies. As inconsistencies are inevitable in an evolving specification but need to be resolved in a final specification, this inconsistency management is an important contributor to the usefulness of the approach.

This viewpoint model has a great deal of flexibility and may be used as a basis for automating part of the requirements analysis process. In our view, however, it suffers from two specific problems which limit the applicability of the method to the later stages of system specification. These two problems are:

1. Its support for inconsistency management relies on the ability of the analyst to write inter-viewpoint consistency rules. Checking these automatically requires the system model to be expressed in at least a structured and, preferably, a formal notation. Where requirements are evolving, this is unlikely to be cost-effective. Specially trained staff are required to write this specialised notation. Furthermore, inconsistencies are often a consequence of interactions between functional and non-functional requirements which are impossible to express in a formal way.
2. It is a 'heavyweight' approach in that it defines a process for establishing and checking the system models and the notations which must be used to establish these models. Although it tolerates inconsistency, it does not tolerate informality nor does the viewpoint structure appear to have any slots for including informal, natural language descriptions of the system model.

In our view, these limitations mean that there is a high introductory cost in introducing this method which constitutes a real barrier to its practical industrial use.

4. Problems with viewpoints

Very few people doubt the wisdom of considering requirements from multiple perspectives. Different models of viewpoints have their own strengths and weaknesses but, without exception, they offer advantages over an unstructured approach to requirements elicitation, definition and specification. However, only CORE has made the transition from research to practice and the use of this method is almost entirely limited to UK defence contractors.

We believe that the technical merits of viewpoint-oriented approaches are self-evident but that the developers of these techniques have paid insufficient attention to the practical problems of introducing them into existing defined and standardised software processes. From our experience in industrial projects, we have identified the following practical problems which make viewpoint-oriented approaches difficult to use for non-trivial projects:

- *Inflexible viewpoint models.* If the viewpoint model is too restrictive in its definition of a viewpoint, it will not encompass all of the possible stakeholder and domain viewpoints which may be required. Many requirements problems are human, social and organisational problems. Viewpoints need to be able to reflect these positions and not just technical expressions of system requirements.
- *Fixed notations for requirements definition.* Requirements sources often will not have time to express requirements in anything but their normal working notations. They are unlikely to be able to translate these easily into some different modelling notation. Automated or semi-automated conflict analysis of requirements is, in our view, impractical.
- *Limited support for requirements evolution.* Not only is the system's organisational, economic and political environment changing as the requirements are developed, the better understanding of the system which emerges during the RE process causes requirements to evolve. Viewpoint-oriented approaches must recognise this and must not, for example, require consistency at all times.
- *Limited support for requirements negotiation.* The process of establishing a final set of requirements for a system normally involves stakeholders negotiating changes and compromises between conflicting requirements. Some means of discovering conflicting and overlapping requirements is helpful here. However, automated conflict resolution may be counter-productive as it does not recognise the non-technical factors which influence the requirements negotiation activity.
- *No industrial-strength tool support.* Viewpoint-oriented approaches tend to generate a large amount of information which must be managed and this obviously requires tool support of some kind. This tool support must be available on platforms used by application developers, must be of good quality, must be compatible with other tools which have already been purchased and must be reasonably cheap. It is extremely difficult to meet these support requirements for any new method because the costs and risks of tool development are so high.

- *No recognition of the problems of non-functional requirements.* In some applications, non-functional requirements are more critical than functional requirements. For example, in many control systems, there is an inflexible requirement to maintain the safety of the system whereas the functional capabilities of the control system are usually negotiable. Requirements engineering methods don't handle non-functional requirements very well, especially when these are 'system-level' requirements rather than requirements associated with a particular function or class of functions.
- *Incompatibility with other software engineering methods.* Organisations which might benefit from the use of viewpoints for requirements engineering generally have existing, defined, design processes. Any requirements engineering method must be compatible with existing design methods.

These are difficult problems and we do not believe that the designers of viewpoint-oriented methods have taken sufficient account of these pragmatic difficulties. If requirements engineering processes are to be improved by introducing viewpoints, then we need a 'lightweight' approach which can be introduced at relatively low cost and risk and which requires evolutionary rather than revolutionary process improvement. Such an approach is described in the remainder of this paper.

5. The Preview approach

The viewpoint model which we describe in the remainder of this paper was developed in a research and development project called REAIMS, whose principal objective was to investigate techniques for requirements engineering process improvement. None of the industrial partners in the project made use of viewpoints explicitly in their requirements engineering processes but all agreed that the approach had potential. However, no existing viewpoint-oriented method could be integrated with their existing processes because of the practical difficulties identified above.

We therefore developed a new model of viewpoints which we call *Preview* (*P*rocess and *r*equirements *e*ngineering *view*points) which could be introduced into existing RE processes in an evolutionary rather than a revolutionary way. The priorities of our industrial partners were to improve the processes of requirements discovery, analysis and negotiation rather than system specification. Preview is therefore classed as a viewpoints approach for requirements elicitation rather than system modelling.

Key characteristics of Preview are:

1. Requirements associated with a viewpoint may be expressed in any notation. Normally, we expect these requirements to be expressed in natural language, tables and diagrams. Structured or formal notations may also be used if appropriate.
2. The analysis is driven by a set of concerns which reflect the critical non-functional characteristics of the system.

3. Viewpoints must limit their scope and explicitly describe their perspective in order to facilitate requirements discovery and analysis.
4. Preview may be used for the analysis of processes as well as system requirements. This is useful because understanding the real requirements for a system is often helped by an analysis of the processes which the system is required to support. We do not cover process analysis here but cover it in a separate paper [Sommerville *et al*. 1995].

5.1. Preview viewpoints

As in other methods, each Preview viewpoint is an entity which encapsulates some but not all information about a system's requirements. This information may be derived from an analysis of existing processes and systems, from discussions with system stakeholders or from domain and organisational information. Complete requirements for the system are created by integrating the requirements derived from different viewpoints.

A Preview viewpoint includes the following information:

1. *The viewpoint name*. This is used to identify and refer to the viewpoint and should normally be chosen to reflect the *focus* of the viewpoint. The name may reflect a role in the organisation or a part of the system or process to which the analysis is restricted.
2. *The viewpoint focus.* A viewpoint's focus defines the scope of the viewpoint. It is expressed as a statement of the perspective adopted by that viewpoint. This is quite difficult to define succinctly and we discuss it in more detail later.
3. *The viewpoint concerns.* The viewpoint concerns reflect the organisational goals, business objectives and constraints which drive the analysis process.
4. *The viewpoint sources.* Viewpoint sources are explicit identifications of the sources of the information associated with the viewpoint.
5. *The viewpoint requirements*. This is the set of requirements arising from analysis of the system from the viewpoint's focus. The requirements may be expressed in terms of system functionality, user needs or constraints arising from application domain or organisational considerations.
6. *The viewpoint history*. This records changes to the viewpoint as an aid to traceability. It includes changes to the focus, the sources and the requirements encapsulated in the viewpoint.

A viewpoint is defined by its focus and we do not exclude any type of viewpoint which an organisation may find useful in its requirements engineering process. However, we have found that viewpoints generally fall into one of three classes:

- *Interactor viewpoints.* These are the viewpoint of something (human or machine) which interacts directly with the system being specified. Examples include human operators who impose usability requirements or requirements for specific process support functions and external systems which impose compatibility and information exchange requirements.
- *Indirect stakeholder viewpoints.* These are the viewpoint of an entity (human, role or organisation) which has an interest (stake) in the problem but who does not interact directly with the system. Examples include operating organisations and standards/regulatory bodies. Indirect stakeholder viewpoints allow Preview to explicitly decouple requirements which might be generated by an operator from those which might be generated by the operator's organisational structure.
- *Domain viewpoints.* These represent a set of related characteristics of the domain which cannot be identified with a particular stake or interactor but which nevertheless impose requirements which are implicit in the domain. For example, requirements on a communication system may be imposed by signal propagation time in copper and optical cables. Because requirements arising from domain phenomena are often part of domain experts' knowledge, they may be overlooked if domain expertise is unavailable or poorly utilised. The use of viewpoints to represent domain phenomena provides a defence against this.

Examples of viewpoints from these classes taken from a computer-based system for engine control in an aircraft might be a *pilot* viewpoint (interactor), a *maintenance planning* viewpoint (indirect stakeholder) and an *electromagnetic radiation environment* viewpoint (domain phenomenon).

The statement of requirements encapsulated in a viewpoint may be expressed in any notation and at any level of detail. This reflects the practical consideration that it is difficult to define requirements precisely at any early stage of the RE process. This lack of precision has benefits as over-prescription at this stage causes problems when conflicts between requirements have to be negotiated.

Of course, this informality does not allow for any automated cross-checking and conflict analysis across viewpoints. We do not feel that the lack of such analysis is a serious problem. We have already discussed that there is a significant cost in writing requirements so that they may be checked and that non-functional considerations are generally excluded from such automated checks. For these reasons, our industrial partners did not consider automated consistency checking to be cost-effective.

Like Nuseibeh *et al.* [1994] we think it useful to maintain a viewpoint change history as an aid to traceability. This is useful during requirements analysis as it helps the analyst to understand changes which have been made and to avoid proposals which have already been rejected during the initial process of requirements discovery. The change history is simply maintained as a textual list of changes made and the associated rationale for these changes.

5.1.1. Viewpoint concerns

Viewpoint concerns are an innovative feature of Preview and have been introduced so that the requirements can be explicitly linked to organisational goals and priorities. Concerns correspond to high-level strategic objectives for the system. They are used to ensure that the requirements for the system are consistent with the business goals of the procuring organisation.

Examples of concerns which are associated with a system might therefore be:

1. *Safety*. Does the system as a whole or parts of the system pose a threat to human life or the system's environment?
2. *Availability*. Will the system be available for service when required?
3. *Functionality*. What functionality must the system provide in order to be of value to the organisation procuring the system?
4. *Maintainability*. What are the implications of specific requirements on the maintainability of the system?

Concerns are established after discussion with strategic management in an organisation and are first expressed at a very high level of abstraction. They are frequently common to applications within the same domain. To be effective, the number of concerns should be small (typically no more than 6) and rigorously scrutinised to eliminate all but the most overriding, system-wide high-level goals and constraints.

It may seem that concerns are a kind of viewpoint but we think it helpful to make a distinction between concerns and viewpoints:

1. Concerns reflect organisational priorities which drive the requirements analysis process.
2. Concerns may be broken down into sub-concerns and finally into specific questions which must be considered by all viewpoints. These questions act as a check list to ensure that requirements from a specific viewpoint do not conflict with organisational priorities.
3. Concerns are a way of expressing critical 'holistic' requirements which apply to the system as a whole rather than to any specific sub-set of its services or functionality. An example of such a requirement, if safety was a concern, might be that software failures cannot propagate across module boundaries. All viewpoints inherit these requirements and viewpoint requirements must not conflict with them.

Concerns cut across *all* viewpoints and the questions associated with concerns must be linked to all viewpoints and posed to viewpoint sources as part of the analysis process. This is illustrated in Fig. 2 which shows classes of possible viewpoints. These range from equipment connected to the system and system operators (at the apex of

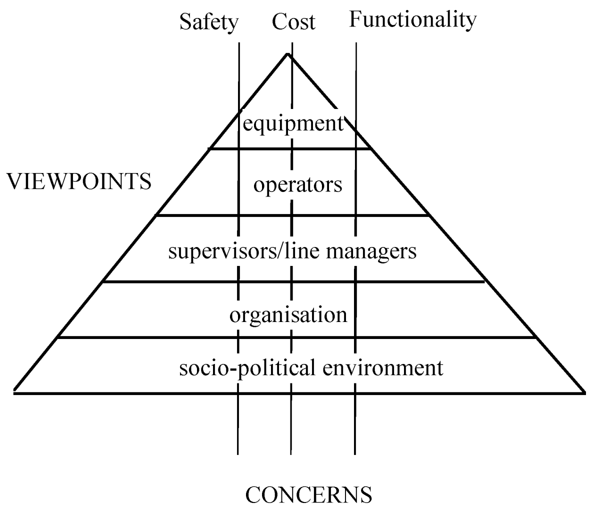


Figure 2. The orthogonality of viewpoints and concerns.

the triangle) to viewpoints associated with the socio-political environment in which the system is installed.

System functionality may be included as a concern like other non-functional concerns such as safety, reliability, performance, etc. This reflects the fact that, for large and complex systems, it is often necessary to make trade-offs between functional and non-functional requirements. System functionality is often negotiable rather than immutable as is sometimes implied by other RE methods.

The practical application of concerns require them to be decomposed into more detailed sub-concerns and questions. Each sub-concern represents a special case of the concern and is applicable to a subset of the problem space. For example, consider a situation where safety is a system concern. In this case, a hazard analysis is carried out to identify hazards which the software system should protect against. Each of these hazards may be expressed as *external requirements* which apply to the system as a whole and are thus part of all identified viewpoints.

Figure 3 shows part of this concern decomposition for a paper guillotine system. This is a software-controlled system where a blade moves to cut stacks of paper to some specified size.

The external requirements (ERx) in this example, are derived from the organisational need for a safe maintenance process. Examples of such requirements are shown in Table 1.

Under the heading of environmental safety, two questions may be identified:

1. In the event of correct operation or system failure, could a requirement compromise the safety of the environment in which the machine is installed?

2. Within a viewpoint, are there any specific requirements for environmental safety which should be identified?

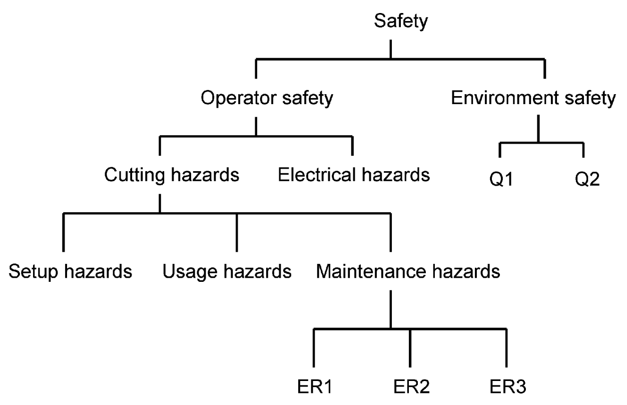


Figure 3. Safety concern decomposition for paper guillotine.

Table 1
External requirements for a guillotine system.

Requirement	Name	Description
ER1	Accidental startup	The control system must disable the guillotine so that it cannot be started accidentally during the maintenance process.
ER2	Blade installation	The control system must include a visual indicator to show that both the principal and backup blade fixing mechanisms are properly secured. The guillotine must be disabled until both principal and backup blade fixing mechanisms are secured.
ER3	Blade removal	The control system must ensure that the blade shield is in place before the blade lock may be opened.

Questions associated with concerns are likely to be fairly general and are used both as a driver of requirements discovery and as a checklist for requirements analysis.

5.1.2. Viewpoint focus

A viewpoint's focus is an explicit statement of the perspective adopted by that viewpoint. The focus can be a statement of the parts of the problem, the system or the process with which the viewpoint is associated, a statement of the role of the viewpoint sources, a statement of an organisational function such as management, health and safety, engineering or, perhaps, a mixture of these. Focus is the defining characteristic of a viewpoint. No two viewpoints have the same foci (otherwise they would by definition be the same viewpoint) but viewpoints may have foci which intersect.

Examples of viewpoint focus might be:

- User requirements concerned with the interactions between human operators and on-board systems

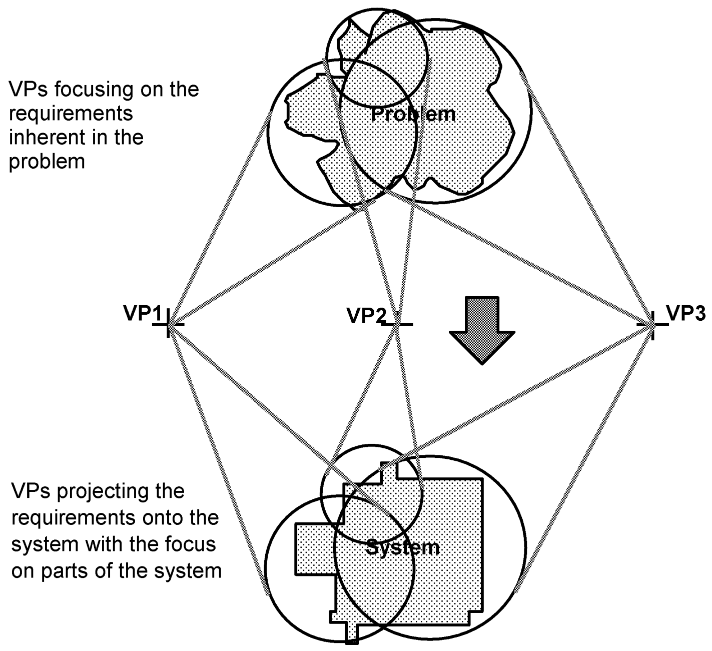

VPs focusing on the requirements inherent in the problem
Problem
VP1
VP2
VP3
VPs projecting the requirements onto the system with the focus on parts of the system
System

Figure 4. Viewpoints on a problem.

- Call charging and customer communication security requirements imposed by a telecommunications regulator
- Constraints imposed on system components affected by proximity to high-voltage ignition system components

Different viewpoints may have overlapping foci (see Fig. 4). For example, in an air traffic control system we may identify a 'chief controller' viewpoint whose focus is the chief controller's interests in the display sub-system. This would clearly overlap with a radar controller viewpoint whose focus was the display sub-system in general. It is important to identify these overlapping foci as they help us discover potential requirements conflicts.

The notion of focus forms a link between the problem space (the "domain" in Jackson's [1995] terms) and the system (the "machine") which is to be developed. Focus is usually defined in such a way that it encompasses both system and domain considerations. This is illustrated in Fig. 4 where the foci of viewpoints VP1–VP3 (represented as circles) cover overlapping parts of both the problem and the system which is to be developed to address the problem.

The notion of viewpoint focus is useful for a number of reasons:

1. It provides a basis for a coverage analysis. By reviewing the foci of viewpoints, we may be able to find specific parts of the system, problem or domain which have not been considered.

2. As we discuss later, it helps identify viewpoints which may include conflicting requirements.
3. It is a useful way of discovering requirements sources, i.e., people or documents which might provide system requirements.
4. Where the focus is primarily domain-based, it serves to identify viewpoints which encapsulate requirements which are potentially reusable across a range of systems.

It is not easy to define, at the outset, the focus of all viewpoints. Focus definition is usually iterative. Initial foci are proposed and requirements sources are identified. A coverage analysis is carried out to see if essential parts of the system or domain are not covered by the union of the viewpoints foci. Along with information from sources, the results of this coverage analysis are then used to re-define the viewpoints' foci.

5.1.3. Viewpoint sources

The sources associated with a viewpoint are an explicit record of where the information associated with a viewpoint has come from. Maintaining such a record is valuable for *external traceability* where a requirement or a group of requirements can be traced to its source. This simplifies analysis when conflicts are discovered and when changes are required.

Viewpoint sources are not simply individuals or roles in an organisation. They may also include:

- manuals of operating procedure;
- international, national or organisational standards;
- domain knowledge;
- experience data such as incident descriptions;
- other requirements placed on the system (requirements often cause further requirements to be generated).

Several sources should be associated with each viewpoint. In a well-understood application, the identification of viewpoint sources normally follows identification of the viewpoint's focus. A source is concrete (a named individual or identifiable document) while a viewpoint's focus may be a role within an organisation, a subsystem or some physical, cognitive or social phenomenon of the domain.

However, in an unfamiliar application domain or organisational structure, the processes of viewpoint identification, focus definition and source selection are interleaved. Here, the set of relevant viewpoints will not be immediately apparent but will emerge as analysis proceeds and domain understanding increases. Having identified a potential viewpoint, the requirements engineer must verify that a corresponding source (e.g., a specific end-user) exists from which the viewpoint requirements may be discovered. Information from that source may then be used to define the focus and hence point to other possible information sources.

Links to sources at the viewpoint level do not link specific requirements to specific sources. We have adopted this coarse-grained traceability for pragmatic reasons in that it reduces the overhead of maintaining the information, it emphasises that sources may be equivalent and it avoids over-burdening the requirements engineering with documentation which must be maintained. Requirements may come from a single source but are an interpretation of information collected from several sources. In this respect, it would be artificial to link a requirement directly to sources.

Of course, we recognise that, for some systems, finer-grain traceability is required. This is particularly likely where there are very specific requirements which are suggested by some source. The Preview model can accommodate this simply by treating a requirement record as a composite document which includes a statement of its sources. The viewpoint sources are then the union of the sources recorded with each requirement.

5.1.4. The Preview process

Preview is applicable to requirements elicitation activities and we have defined an informal process which may be used to apply the Preview approach. This is a 6-step process as illustrated in Fig. 5.

5.1.5. Requirements discovery

The requirements discovery process is broken down into four activities namely:

- identification of concerns;
- elaboration of concerns as external requirements and questions;
- identification of viewpoints;
- discovery of the requirements for each viewpoint.

Concern identification and elaboration are critical activities as the fundamental role of concerns is to concentrate the requirements elicitation process on the factors which are central to the system's success. They correspond to very high-level strategic goals of the organisation procuring and/or developing the system. The first phase in their identification is to ask the question:

What fundamental properties must the system exhibit if it is to be successful?

Where 'successful' may mean several things; e.g., profitable for the developer, allowing the customer's market position to be maintained, satisfying the current and projected operating regulations, etc. Knowledge of the fundamental principles of the application domain and the constraints under which the stakeholders operate are necessary for concern identification. Concerns are therefore normally elicited from the customer (or developer's marketing organisation) and the developer of the system.

Examples of answers to the above question might be:

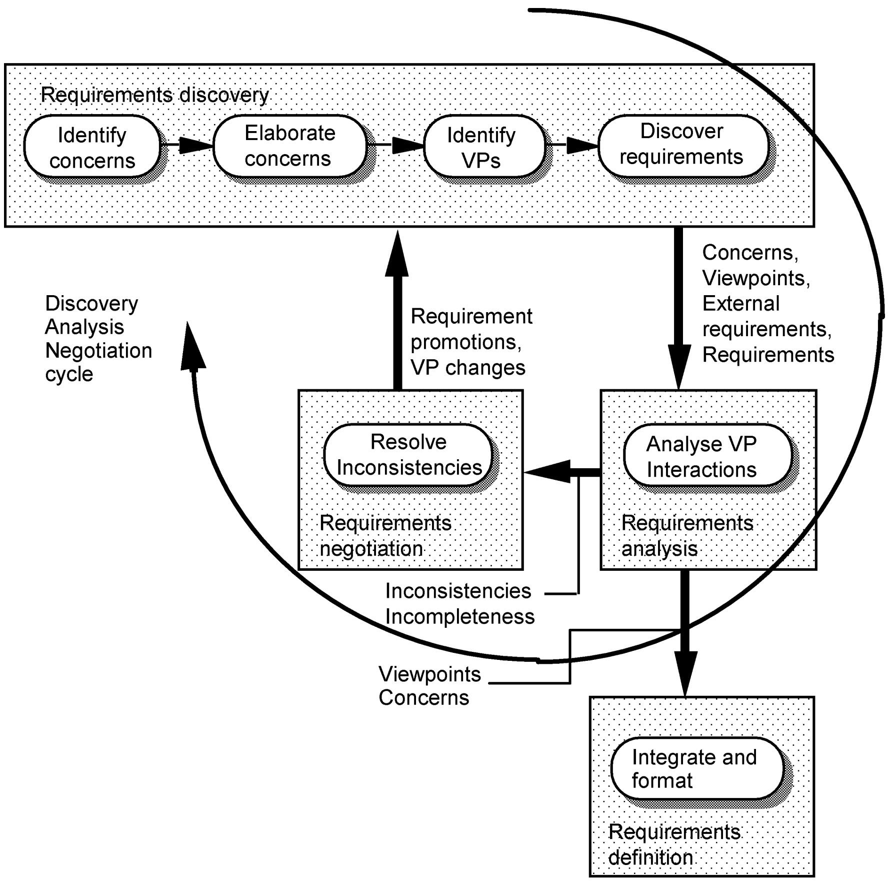


Figure 5. The Preview process.

Application 1: Bank customer account system
 Stakeholder: Bank
  Security, Availability
 Stakeholder: Developer
  Compatibility

In this example, the stakeholders are the bank and the software developer. The bank will suffer a catastrophic loss of reputation if the public perceives their systems to be insecure so security is a concern. Similarly, the public's tolerance of poor availability will be low so this too is a concern. The software developer will only make money if development and maintenance costs are minimised and these will be high if the issue of compatibility with existing bank systems is not explicitly addressed.

Application 2: Anti-lock braking system (ABS) control software
 Stakeholder: Car manufacturer's marketing division
  Safety, Functionality

Stakeholder: Component supplier
Reuse

In this example, the stakeholders are a car manufacturer (as represented by their marketing division) and a component supplier. As with the bank, the car manufacturer is acutely sensitive to reputation. Hence, its concerns are that its ABS system is perceived to be safe and, in order to give a competitive edge, more functional than competitors' products. The component supplier is here concerned with maximising the reuse of existing components to minimise development costs.

Once identified, the concerns must be elaborated so that they can exert a direct influence on subsequent requirements activities. The decision on whether to elaborate them to external requirements or concern questions depends, at least partly, on the extent to which the concern can be made explicit.

For example, as illustrated earlier a safety concern can typically be decomposed to a number of external requirements, each associated with a specific hazard identified by hazard analysis techniques. Similarly, a security concern could be decomposed to a number of external requirements, each associated with the avoidance of a particular security risk. An availability concern could be decomposed to a specific external requirement expressed in terms of some availability metric.

In the case of the availability concern, a specific requirement can be identified; for safety and security, this may be more difficult so associated questions are derived to assess requirements against these concerns.

Following concern identification and elaboration, the application must be analysed to identify the set of viewpoints to be used. This analysis is based on an iterative process where viewpoints are proposed and their foci and sources identified. These are analysed. The set of viewpoints is then re-examined to assess if new viewpoints are required, if viewpoints are redundant or if the focus of identified viewpoints should change. Iteration continues until a stable set of viewpoints with defined foci and sources have been identified.

This identification process is a judgmental one which must be based on organisational and application domain experience. However, we do provide some guidance with initial viewpoint identification. This is based on a viewpoint hierarchy which is based on a decomposition of viewpoint types. The initial viewpoint hierarchy which we propose is included in Appendix A but we recommend that organisations tailor and adapt this to their own requirements.

The identification guidelines which we use are:

1. At least three viewpoints should be identified corresponding to an interactor viewpoint, an indirect stakeholder viewpoint and a domain viewpoint.

2. Using the organisational viewpoint hierarchy, decide which viewpoints are likely to be relevant to the problem. Prune unwanted viewpoints from the hierarchy.

3. If more than 6 viewpoints remain, think carefully about whether all viewpoints are really necessary. Too many viewpoints at this stage can lead to an explosion

of information which must be managed and an unduly expensive elicitation process.

4. Define the foci of each identified viewpoint. If this is difficult or unduly vague, you probably need to define more specific viewpoints.

After an initial set of viewpoints have been identified, their foci should be compared. This may identify where gaps lie in the viewpoint requirements' coverage of the system functions and constraints. The discovery of these gaps – for example, if no viewpoint imposed requirements on the characteristics of a database server to be used in the bank customer account system – implies that the resulting requirements specification will be incomplete.

New viewpoints may have to be added to the existing set, whose foci explicitly included the missing areas. They are emergent viewpoints, the need for which only become apparent following analysis of the existing viewpoints' foci. The use of focus analysis cannot, of course, guarantee completeness, but it acts as a mechanism to uncover viewpoints not immediately apparent from the initial analysis of the domain.

Once the set of viewpoints has been defined, the requirements discovery process may begin. This may involve the development of outline system models from background material, structured interviews with sources, observations of processes, analyses of data and data processing, etc.

In the requirements discovery process, we try to avoid 'blank sheets'. If we ask people what they want without presenting them with some alternatives, they may either express unrealistic requirements or have little idea where to start. Rather, we start with an initial outline of requirements and ask sources to describe its deficiencies and omissions. As sources are consulted, the requirements description is refined. It is therefore important to have several sources associated with each viewpoint.

This approach helps to reduce intra-viewpoint conflicts as many inconsistencies and omissions are immediately revealed. Source B is presented with source A's requirements and expresses their requirements as an extension to these. If conflicts arise, we try to resolve them by informal negotiation. If this fails, the conflict is recorded and taken forward for later analysis and negotiation.

It is sometimes helpful to partition requirements which have been discovered by decomposing an existing viewpoint into sub-viewpoints and associating relevant requirements with each of them. Decomposition of a viewpoint into sub-viewpoints should be considered if:

- *The internal requirements lack cohesiveness.* If the internal requirements apply to disjoint sub-sets of the viewpoint's focus, this implies distinct viewpoints. For example, there may be fare collection and vehicle control sub-viewpoints where a viewpoint is associated with the operator of some public transport system.
- *The internal requirements conflict.* Internal requirements may conflict, especially if they are derived from different sources. In itself this is not necessarily sufficient to justify viewpoint decomposition. It is common for two users to

articulate different requirements even if their roles are the same. However, if the sources of the requirements have imperfectly matched foci then viewpoint decomposition may be appropriate.

The cost of adding new viewpoints at this stage is much less than identifying them initially as they are a structuring mechanism for the requirements rather than a way of organising the elicitation process. Of course, simply decomposing a viewpoint with conflicting requirements doesn't reconcile them but it helps the requirements negotiation process by partitioning the problem and associating conflicting requirements with their source.

5.1.6. Requirements analysis

The purpose of this phase of the Preview process is to identify those requirements which are inconsistent with concern questions, external requirements, or other viewpoints' requirements. The objective is to discover internal viewpoint conflicts and external inconsistencies where requirements from different viewpoints are in conflict. The result of this activity acts as the input to the requirements negotiation activity where the inconsistencies should be resolved.

Internal conflicts and omissions are highlighted by analysing each viewpoint requirement against the concerns which drive the analysis. Hence for concern questions, those questions should be posed for each requirement, e.g., "*What are the safety implications of this requirement*?." Similarly, where concerns have been decomposed to external requirements, each of these must be checked for compatibility with the viewpoint's requirements. All sources associated with a viewpoint should review the viewpoint requirements to discover potential inconsistencies.

When all viewpoints are internally consistent, the requirements in each viewpoint must then be checked for external consistency. This consistency checking does not simply encompass conflicting functional requirements. As well as being checked against each other, functional and non-functional requirements must also be checked against domain and organisational requirements.

For example, a *user* viewpoint on a network operating system might include a functional requirement that files' locations can be changed, and a non-functional requirement that drag-and-drop should be the user's normal means of interacting with the file store. A *network* viewpoint may be used to represent constraints and requirements imposed on the system by the network such as the rate of data transmission and protocols used. When analysed with the requirements from the user viewpoint, a potential inconsistency may become apparent. Drag-and-drop interaction, with its reliance on rapid semantic feedback, may be impractical due to network delays.

Preview's approach is to exploit the focus attribute to direct the requirements engineer's attention to viewpoints which impose requirements on the same system components or features. Viewpoints whose foci intersect are the most likely sources of conflict. Figure 6 illustrates a scenario with 3 viewpoints; viewpoints VP1 and

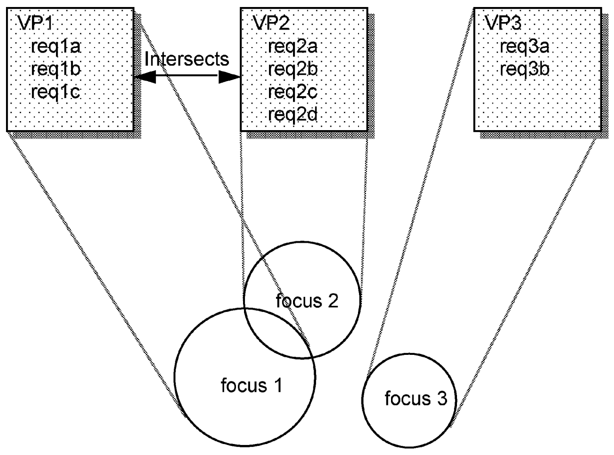


Figure 6. Viewpoints with overlapping and non-overlapping foci.

VP2 have intersecting foci so the three requirements of VP1 should be checked for consistency with the four requirements of VP2.

By comparing viewpoints' foci and isolating only those whose foci intersect, the requirements engineer narrows the search space for inconsistent requirements. VP3's focus is disjoint from both VP1 and VP2 so there is less chance of conflict between these viewpoints. Of course, there could be some subtle conflicts which are not reflected in the focus or there could be unforeseen overlaps between the foci. Nevertheless, the explicit guidance provided by overlapping foci means that limited analysis effort can be deployed in the most effective way.

For each pair of viewpoints with intersecting foci, their requirements must be checked for mutual consistency. We propose a simple tabular method similar to the Quality Function Deployment (QFD) method [Errikson and McFadden 1993] to act as a checklist of requirements' compliance. Table 2 shows a tabular plotting of the intersecting viewpoints VP1 and VP2 from Fig. 6. Here, VP1's requirements are represented as rows and VP2's requirements are represented as columns. Where they intersect, the requirements engineer must examine them to assess whether they are:

- *Overlapping*. There is some overlap between the requirements in each viewpoint which should be discussed with a view to simplifying the requirements. A '1000' is used to indicate overlapping requirements.
- *Conflicting*. There is a conflict between the two requirements which should be resolved. A '1' is used to indicate conflicting requirements.
- *Independent*. The viewpoint requirements are independent. A '0' is used to indicate two independent requirements.

In this example, req1b conflicts with req2a and req2c, while req1c conflicts with req2d. These conflicts would go forward to the requirement negotiation phase for resolution. Similarly, req1b and req1c overlap with req2b. When the requirements

Table 2
Checking VP1 and VP2 for mutual consistency.

		VP2			
		req2a	req2b	req2c	req2d
VP1	req1a	0	0	0	0
	req1b	1	1000	1	0
	req1c	0	1000	0	1

specification document is developed from the viewpoints analysis, it may be possible to rationalise these three requirements.

The advantage of using numeric values to represent overlapping and conflicting requirements is that a simple spreadsheet may be used to assist with the analysis process. By summing the rows and columns, dividing the result by 1000 and taking the dividend and the remainder, we can identify the number of overlaps and conflicts. The most problematic requirements are revealed and may be given particular consideration in the requirements negotiation process.

We recognise that using the viewpoint focus to discover potential conflicts is not an infallible process. Some kinds of conflict will not be revealed by this approach particularly those arising from implicit organisational and political factors. For example, bank staff may deliberately design procedures for customer interaction so that they cannot be readily automated. They may then express automation requirements which they know are unrealisable so that the system development fails and they keep their jobs. We are not aware of any approach which addresses this type of problem.

5.1.7. Requirements negotiation

The inputs to this phase are the sets of conflicting and overlapping requirements. Preview does not prescribe how conflicts are resolved or how overlapping requirements are rationalised as this will typically necessitate trade-offs and compromise. Normally, we expect the people who are viewpoint sources to meet together, discuss the requirements and agree on priorities. The viewpoint concern, focus and source information associated with each requirement should be used to inform the negotiation process and provide a context for resolution.

The results of the process will typically be a set of changed requirements. These changes should be fed back to the requirements discovery phase and should be recorded in the relevant viewpoints' histories.

It is feasible that some requirements in conflict with others prove, on analysis, to always take precedence or to be immutable. The identification of immutable requirements reveals that their status is on a par with that of concerns and should be treated as such. Consideration should be given to 'promoting' these to associate them with concerns and thus ensure that they are considered by all viewpoints in subsequent analysis. This feedback connection provides some measure of self-correction in the process to guard against a failure to identify all pertinent concerns in the early stages of the requirements discovery phase.

6. Conclusions

Preview is a pragmatic framework for requirements elicitation activities. It can be introduced into real RE processes in an evolutionary rather than a revolutionary way.

The Preview approach is distinguished by the following characteristics:

1. The approach explicitly recognises the importance of organisational needs and priorities through its identification of concerns. The Preview process has been designed to include these explicitly in the analysis.
2. The lack of prescription in the viewpoint model means that it can be used with existing design and analysis notations. Viewpoints do not have to conform to a particular type so the applicability of the approach is generic rather than specific to a particular class of system.
3. Viewpoints are independent entities which do not rely on explicit relationships or the existence of other viewpoints. The approach may be used incrementally. Adding more viewpoints improves coverage but benefits accrue when as few as two viewpoints are explicitly recognised.
4. Preview supports a user-centred approach by recommending that interactor viewpoints should always be considered in the requirements elicitation process.

A key goal of the development of Preview was that practical industrial requirements should be taken into account. Table 3, shows our assessment of Preview against the practical problems of introducing viewpoint-oriented approaches identified in section 5.

The Preview approach has been applied in a number of projects in the aerospace and railway signalling domain. The key results of these projects confirmed our views that flexibility is the key to practical use of viewpoint-oriented approaches. Some conclusions of the evaluation were:

1. An initial viewpoint decomposition such as that given in the appendix is useful but too many viewpoints result in an unmanageable explosion of information. Therefore, after the initial decomposition (which is very helpful for identifying viewpoint sources), there should be a reification step to reduce the number of viewpoints. We suggested six as the maximum – in practice, we found that no-one wanted to work with more than four viewpoints.
2. An important advantage of viewpoints was that it structured the presentation of requirements. It helped make clear where requirements came from (e.g., a safety requirement) and, in general, made the requirements more readable.
3. It helps to have both viewpoints and concerns but there is a blurred distinction between them. It is sometimes necessary to turn viewpoints into concerns and vice-versa. A pragmatic way which we found to address this problem was to allow a concern to be another viewpoint.

Table 3
Preview assessment.

Identified problem	Preview assessment
Inflexible viewpoint models	Preview viewpoints are flexible and user-definable. They are defined by their focus rather than according to any pre-defined model. They may be established according to organisational needs and may range from system end-users to organisational, social and political constraints on the system.
Fixed notations for requirements definition	There is no pre-defined notation for expressing requirements. Any appropriate notation may be used. This includes natural language, diagrams, equations, formal descriptions or graphical system models.
Limited support for requirements evolution	Backward traceability is essential for supporting evolution. Preview viewpoints include links to sources of requirements and maintain change histories.
Limited support for requirements negotiation	Problems are most likely to arise when viewpoints have overlapping foci. The QFD-based comparison identifies overlapping and conflicting requirements which may be an input to the negotiation process. The explicit definition of a viewpoint focus also helps with viewpoint prioritisation.
No industrial-strength tool support	The method is not reliant on special-purpose tool support. It has been designed so that existing tools for requirements management may be used to manage the information in a viewpoint.
No recognition of the problems of non-functional requirements	We have addressed this, to some extent, with the notion of concerns which ensures that critical non-functional requirements may drive the analysis process and all identified requirements are checked against them.
Incompatibility with other software engineering methods	This problem has not been explicitly addressed. As the system does not depend on pre-defined notations, requirements modelling notations which are compatible with design notations may be used. Therefore, if an organisation is already committed to object-oriented development, viewpoints may include object models which are processed by existing tools.

4. Conflict analysis was not as much of a problem as we had envisaged. Viewpoints had limited but well-defined areas of overlap and there was no need to use the QFD-based approach which we had proposed. This may reflect the class of system used as examples and the fact that the stakeholders had a fairly clear idea of their requirements. We suspect that, for information systems, this may not be true.
5. Almost everything had to be interpreted flexibly and introduced incrementally. For example, our viewpoint identification guidelines (find an interactor, an indirect stakeholder and a domain viewpoint) were largely ignored. Users based viewpoint identification on their own experience. Furthermore, if there is no history in an organisation of recording requirements sources, it is very hard to get people to include this information. This doesn't mean it isn't useful – it simply means that time is needed to change working practices.

We are investigating possible extensions to the viewpoint concept to improve its usability, carrying out further trials on real applications and exploring how tool support for the approach may be provided. We are completely against the notion of special-purpose tools and are committed to evolving the concept, if necessary, so that it can be used with existing tools for requirements management. Our initial experiments in this are based on a tool called DOORS (see URL http://www.qss.co.uk/ for more information) and it seems that this system can support most of the viewpoint model which we have developed.

A possible extension is the notion of a viewpoint vocabulary. Terminology problems are endemic in requirements engineering and it seems to us that defining a viewpoint vocabulary would create a reusable resource which could be used to discover when sources were using the same terms in the same way. However, we have not yet explored how this concept may be supported and used.

In summary, Preview helps improve the quality of requirements specification by providing a framework which can support both requirements elicitation and the structuring of the requirements document. In some respects, it is 'less advanced' than other approaches. We do not propose technological solutions such as automated conflict analysis nor have we invented expressive notations or new processes. However, we have built on previous research to develop an evolutionary approach which addresses outstanding requirements engineering problems in a way which is pragmatic and relevant to industrial needs.

References

Barlas, S. (1996), "Anatomy of a Runaway: What Grounded the AAS," *IEEE Software 13*, 1, 104–106.

Booch, G. (1994), *Object-Oriented Analysis and Design with Applications,* Benjamin Cummings, Redwood City, CA.

Bowman, H., J. Derrick, P. Linington, and M. Steen (1996), "Cross-viewpoint Consistency in Open Distributed Processing," *BCS/IEE Software Engineering Journal 11,* 1, 44–57.

Easterbrook, S. and B. Nuseibeh (1996), "Using ViewPoints for Inconsistency Management," *BCS/IEE Software Engineering Journal 11,* 1, 31–43.

Errikson, I. and F. McFadden (1993), "Quality Function Deployment: A Tool to Improve Software Quality," *Information and Software Technology 35,* 9, (pages?).

Feather, M. (1993), "Requirements Reconnoitering at the Juncture of Domain and Instance," In *Proceedings of the International Symposium on Requirements Engineering,* IEEE Press, Los Alamitos, CA, pp. 73–77.

GAO (1979), "Contracting for Computer Software Development – Serious Problems Require Management Attention to Avoid Wasting Millions," US General Accounting Office.

Gibbs, W.W. (1994), "Software's Chronic Crisis," *Scientific American 271,* 3, 72–81.

Greenspan, S. and M. Feblowitz (1993), "Requirements Engineering Using the SOS Paradigm," In *Proceedings of RE'93,* IEEE Press, Los Alamitos, CA, pp. 260–265.

Hughes, J., J. O'Brien, T. Rodden, M. Rouncefield, and I. Sommerville (1995), "Presenting Ethnography in the Requirements Process," In *Proceedings of RE'95,* IEEE Press, Los Alamitos, CA, pp. 27–35.

Jackson, D. and M. Jackson (1996), "Problem Decomposition for Reuse," *BCS/IEE Software Engineering Journal 11,* 1, 19–30.

Jackson, M.A. (1983), *System Development,* Prentice-Hall, London, UK.

Jackson, M.A. (1995), "Problems and Requirements," In *Proceedings of the 2nd IEEE International Symposium on Requirements Engineering,* IEEE Press, Los Alamitos, CA, pp. 2–9.

Jacobson, I., M. Christensen, P. Jonsson, and G. Overgaard (1993), *Object-Oriented Software Engineering*, Addison-Wesley, Wokingham, UK.

Jarke, M., J. Bubenko, C. Rolland, A. Sutcliffe, and Y. Vassiliou (1993), "Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis," In *Proceedings of the IEEE Symposium on Requirements Engineering*, IEEE Press, Los Alamitos, CA, pp. 19–31.

Klein, M. (1992), Conflict Management in Concurrent Engineering (special issue), *Concurrent Engineering Journal 2*, 3.

Kotonya, G. and I. Sommerville (1992), "Viewpoints for Requirements Definition," *BCS/IEE Software Engineering Journal 7*, 6, 375–387.

Kotonya, G. and I. Sommerville (1993), "A Framework for Integrating Functional and Non-Functional Requirements," In *Proceedings of the IEE Workshop on Systems Engineering for Real-Time Applications*, IEE, Stevenage, UK, pp. 148–153.

Kotonya, G. and I. Sommerville (1996), "Requirements Engineering with Viewpoints," *BCS/IEE Software Engineering Journal 11*, 1, 5–18.

Leite, J.C.S.P. and P.A. Freeman (1991), "Requirements Validation through Viewpoint Resolution," *IEEE Transactions on Software Engineering 17*, 12, 1253–1269.

Linington, P.F. (1995), "RM-OD – The Architecture," In *Proceedings of the IFIP TC6 International Conference on Open Distributed Processing*, North-Holland, Amsterdam, The Netherlands, pp. 118–126.

Moyse, R. (1992), "VIPER: The Design and Implementation of Multiple Viewpoints for Tutoring Systems," In *Knowledge Negotiation*, R. Moyse and M.T. Elsom-Cook, Eds., Academic Press, London, UK, pp. 93–107.

Mullery, G. (1979), "CORE – A Method for Controlled Requirements Specification," In *Proceedings of the 4th International Conference on Software Engineering*, IEEE Press, Los Alamitos, CA, pp. 126–135.

Nuseibeh, B., J. Kramer, and A. Finkelstein (1994), "A Framework for Expressing the Relationships between Multiple Views in Requirements Specifications," *IEEE Transactions on Software Engineering 20*, 10, 760–773.

Ross, D.T. (1977), "Structured Analysis (SA). A Language for Communicating Ideas," *IEEE Transactions on Software Engineering SE-3*, 1, 16–34.

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen (1991), *Object-Oriented Modeling and Design*, Prentice-Hall, Englewood Cliffs, NJ.

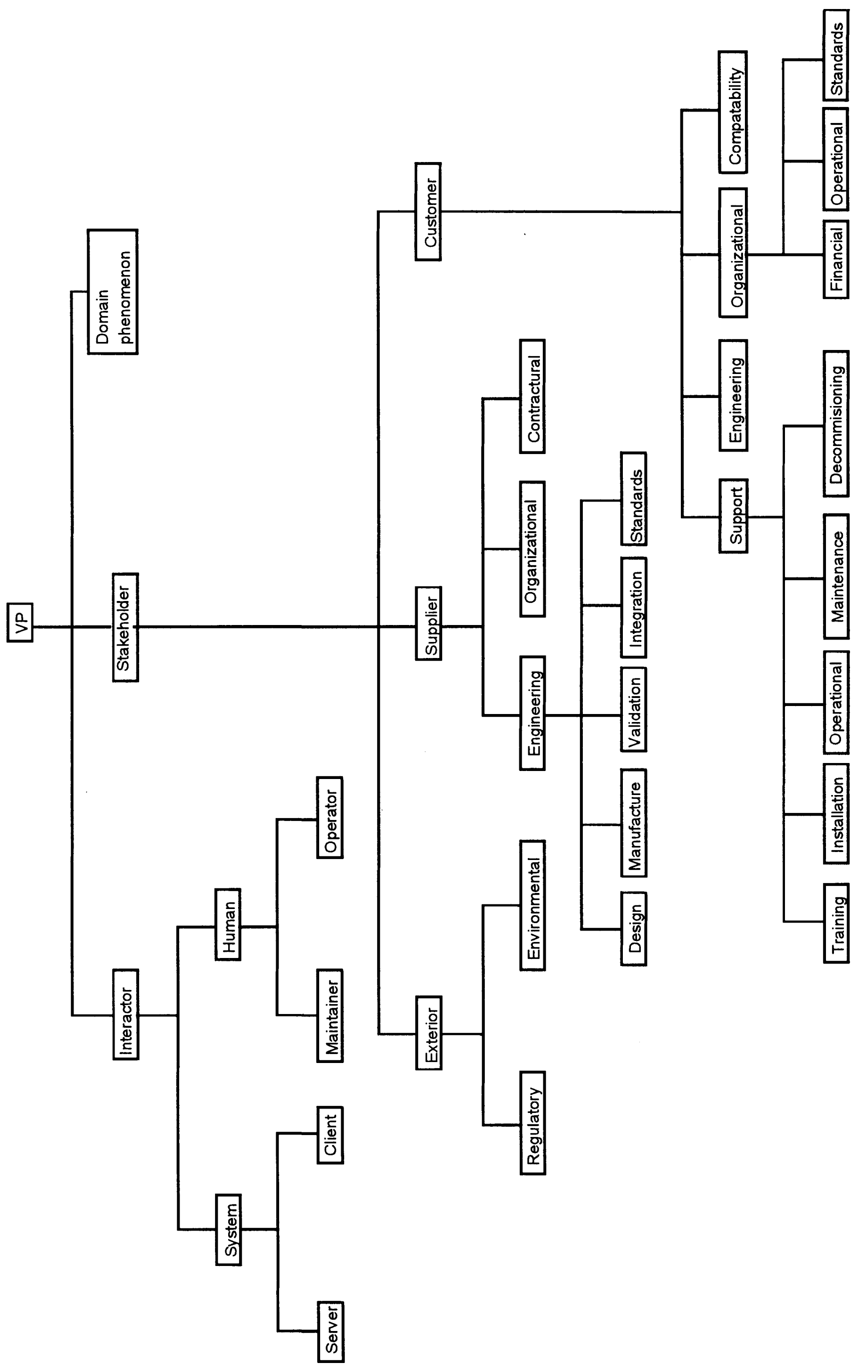
Schoman, K. and D.T. Ross (1977), "Structured Analysis for Requirements Definition," *IEEE Transactions on Software Engineering SE-3*, 1, 6–15.

Self, J. (1992), "Computational Viewpoints," In *Knowledge Negotiation*, R. Moyse and M.T. Elsom-Cook, Eds., Academic Press, London, UK, pp. 54–68.

Sommerville, I., P. Sawyer, G. Kotonya, and S. Viller (1995), "Process Viewpoints," In *Proceedings of the 5th European Workshop on Software Process Technology*, Springer, Berlin, Germany.

Zemanek, G., H. Huber, H. Nissen, and M. Jarke (1995), *Requirements from Multiple Perspectives: Experiences with Conceptual Modeling Technology*, Lehrstuhl für Informatik, Aachen, Germany.

Appendix


VP
Domain phenomenon
Stakeholder
Interactor
Human
Operator
Maintainer
System
Client
Server
Customer
Supplier
Exterior
Contractural
Organizational
Engineering
Environmental
Regulatory
Standards
Integration
Validation
Manufacture
Design
Compatability
Organizational
Engineering
Support
Standards
Operational
Financial
Decommisioning
Maintenance
Operational
Installation
Training