Requirements Eng (1999) 4:115–133

Requirements Engineering

Practical Experience with Viewpoint-Oriented Requirements Specification

G. Kotonya
Computing Department, Lancaster University, UK

The notion of viewpoints as a means of eliciting and formulating requirements is now well known. However, there is little practical evidence that viewpoint-based requirements methods scale up to address real problems. This paper presents a detailed case study based on a medium-sized system, and illustrates how a viewpoint-based requirements method can be used to structure and specify system requirements. The case study is intended to serve two purposes: first, to demonstrate the scalability of viewpoint-based requirements methods; and second, to act as a shared example for other researchers in the field to test their techniques and methods. The case study is based on an electronic document delivery and interchange system (EDDIS). The requirements are presented as they appeared in the original user requirements document. The paper concludes by outlining the lessons learnt in applying VORD to EDDIS, and proposes a set of 10 comparators that other researchers can use to compare their approaches and techniques.

Keywords: Case study; Experience; Requirements definition; Requirements specification; Viewpoints

1. Introduction

The notion of viewpoints as a means of formulating software requirements is well documented [1–5]. The proposed approaches range from those based on extensions to structured analysis, such as CORE [6], to those that use viewpoints as a vehicle for eliciting and specifying requirements [2], through to those that use viewpoints as means for integrating multiple techniques in system development [7]. Some work has also been done on the development of meta-viewpoint techniques; the Preview method is a good example of this [8]. However, apart from the CORE method [6], there is little published evidence that these methods scale to handle real requirements problems.

The case study described here can be viewed as a detailed real version of a library example such as are commonly used in literature [5,9,10]. The simple example, whilst useful for illustrating different requirements engineering techniques, suffers from two main problems:

- It does not reflect the requirements of a real library system.
- Its lack of detail makes it difficult to demonstrate the scalability of the techniques.

In this paper we take the notion of a shared example further and provide a more extensive but understandable example which:

- Presents more than 30 requirements of a real library system on which other researchers in the area can test their techniques.
- Demonstrates how a viewpoint-oriented requirements method, VORD, can be used to specify the system.

The system described here is an electronic document delivery and interchange system (EDDIS). EDDIS is an Electronic Library Programme (ELIB) sponsored project whose objective is to develop a web-based library system for the United Kingdom Higher Education sector. Although the initial proposal for EDDIS was conceived in the context of delivery of documents electronically, EDDIS is also required to manage the request and supply of printed documents across libraries, and is not restricted to digitised documents.

Correspondence and offprint requests to: Gerald Kotonya, Computing Department, Lancaster University, Lancaster LA1 4YR, UK. E-mail: gerald@comp.lancs.ac.uk

The rest of the paper is organised as follows: Section 2 provides a brief description of EDDIS and the requirements for the system. Section 3 describes VORD and how it was used to develop the EDDIS requirements. Section 4 discusses the lessons learnt and provides some suggestions on how the problems can be tackled. Section 5 provides a summary of the work.

2. EDDIS

The EDDIS system intends to support electronic document delivery across a consortium of university libraries. Users can request documents held remotely and read, print and store them at their local university.

The requirements for EDDIS can be summarised thus:

- Identifying documents
- Locating documents
- Ordering documents
- Receiving documents (digitised and non-digitised)
- Providing access to documents received
- Managing the receipt, lending and return of non-digitised documents
- Acting as a supplier in response to orders for documents from other agents (including other EDDIS servers)
- Keeping track of appropriate management account information

Figure 1 shows the EDDIS context diagram with the main stakeholders.

2.1. System Requirements

In this section we describe a subset of the EDDIS requirements exactly as they appeared in the original user requirements document. This document is, we believe, written in the same way as many real requirements documents, where requirements are written as numbered paragraphs of natural language.

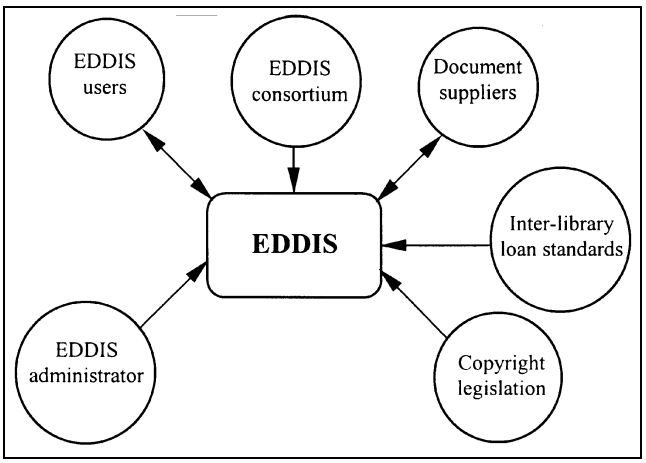

Fig. 1. Context diagram for EDDIS.

1. EDDIS will provide a mechanism for the management of ordering and supplying of all types of documents, both digitised and non-digitised. In the case of the receipt of non-digitised documents, EDDIS will receive and manage both returnable and non-returnable items, that is, EDDIS will have a circulation system with the following features:
   (i) Lending documents
   (ii) Returning documents
   (iii) Recalling documents
   (iv) Renewing documents
   (v) Dealing with overdue documents and processing fines.
2. In addition to requesting documents, EDDIS will manage incoming requests for documents from other agents, including other EDDIS servers, that is, it will act as a supplier.
3. EDDIS will provide a range of services available only to the system administrator. These are primarily:
   (i) Specific administrative services, for example, the registration of EDDIS users.
   (ii) Management of non-digitised documents.
4. EDDIS will be configurable so that it will comply with the requirements of all UK and (where relevant) international copyright legislation. Minimally this means that EDDIS must provide a form for a user to sign the copyright declaration statement. It also means that EDDIS must keep track of copyright declaration statements, which have been signed/not signed. Under no circumstances must an order be sent to supplier if the copyright statement has not been signed.
5. The prototype EDDIS will be required to be operational at the four development sites by 1 March 1997.
6. Interface requirements:
   (i) The user interface will be based on the hypertext mark-up language (HTML). Users will access EDDIS via standard Web browsers.
   (ii) EDDIS will be primarily an end-user system. Users will use the system within the constraints of the permissions assigned by the administrator; to identify, locate, order and receive documents.
7. Accounts:
   (i) Users will log onto EDDIS via accounts, which will be created by the administrator. There will be two types of accounts: individual and group accounts. In general, individual accounts will have access to more services than group accounts.

(ii) Individual accounts are intended for single users. All individual accounts will be password protected. Users of these accounts will be able to change the passwords in their accounts. Group accounts are intended for groups of users, for example, members of the institution, faculty and department. Some of these will be password protected, others will not. However, only the administrator will be able to change group account passwords.

8. Services:
   (i) Users will have access to a range of services determined by the permissions associated with the accounts they use. The administrator will set the permissions. Services available within accounts will vary: some accounts will have access to most of the EDDIS services, but others will be severely restricted. For example, some accounts will be able to search all databases available to EDDIS, also to locate and order documents; whereas others might only be able to search a restricted set of databases and not be able to order documents.
   (ii) There will be four primary services available to users:
   - Document search
     Will allow users to search for and identify documents, which interest them. A document search will be initiated by search criteria and the output will be a set of document-ids which will act as input for document locate and order services.
   - Document locate
     Will allow users to determine the location of documents. A document locate will be initiated by a set of document-ids and the output will be a set of location-ids.
   - Document order
     Will allow users to order documents. A document order will be initiated by a set of document-ids and location-ids. The output will initially be a set of order-ids and eventually the documents.
   - Document read
     Will allow users to read and where appropriate, print documents.

   (iii) There will be various secondary services:
   - Status enquiry
     Will allow users to check the status of document orders.
   - User statistics
     Will provide the administrator with information about the performance and use of EDDIS
   - Non-digitised documents
     Will allow access to non-digitised documents: books, photocopies, films, fiche etc.

9. Communication:
   (i) Users will communicate with EDDIS mainly via the HTML interface.
   (ii) User input to EDDIS will be via the HTML interface.

3. Viewpoint-Oriented Requirements Definition Method

This section provides an overview of the Viewpoint-Oriented Requirements Definition method (VORD) [2]. VORD is based on viewpoints that focus on user issues and organisational concerns. The model adopted for viewpoints is service-oriented where viewpoints are analogous to clients in a clientserver system. The system delivers services to viewpoints and the viewpoints pass control information and associated parameters to the system. Viewpoints map to classes of end-users of a system or to other systems interfaced to it. The viewpoints that make up the core model are known as *direct viewpoints*. To allow for organisational requirements and concerns to be taken into account, viewpoints concerned with the system's influence on the organisation are also considered. These are known as *indirect viewpoints.*

A VORD viewpoint is an entity outside the system that generates a requirement (i.e. a requirement source). It can be a system user, a sub-system interfaced to the intended system or an organisational concern. Viewpoints are structured into a classification hierarchy to accommodate the variations in user requirements.

VORD viewpoints fall into two classes:

1. *Direct viewpoints* correspond directly to clients in that they receive services from the system and send control information and data to the system. Direct viewpoints are either system operators/users or other sub-systems, which are interfaced to the system being analysed.
2. *Indirect viewpoints* have an 'interest' in some or all the services which are delivered by the system but do not interact directly with it. Indirect viewpoints may generate requirements, which constrain the services delivered to direct viewpoints, and the system development process.

Indirect viewpoints vary radically. They may include engineering viewpoints (i.e. those concerned with the system design and implementation), organisational viewpoints (those concerned with the systems influence on the organisation) and external viewpoints (those

concerned with the system's influence on the outside environment). Therefore, if we take the example of a library system, an indirect viewpoint might be a trade-union viewpoint that is concerned with the effects of system introduction on staffing levels and library staff duties.

The importance of stakeholder perspectives in formulating and reconciling requirements is also recognised in other approaches. The ORDIT [11] approach allows system designers to reason about organisational goals, policies and structures and the work roles of intended end-users in a way which facilitates the identification and expression of requirements for information systems. Boehm [12] proposes a spiral approach that revolves around identifying stakeholders, their 'win' conditions and reconciling the 'win' conditions to establish the next-level objectives, constraints and alternatives. Dardenne [13] proposes a goal-directed approach for model acquisition. Goals are seen as determining the respective roles of agents in the system and providing a basis for defining which agents should best perform which actions. Although these approaches may have some similarities with viewpoint methods, their notion of viewpoint is less explicit. Therefore, we do not regard them as viewpoint-oriented.

In VORD, viewpoints and requirements may be described using standard templates. A viewpoint template comprises the components shown in Fig. 2.

A requirement template comprises the information shown in Fig. 3.

Figure 4 shows how the viewpoint and requirement components are related. The main distinction between a direct and indirect viewpoint is in the generated requirements. Unlike a direct viewpoint, which may generate both functional non-functional requirements, an indirect viewpoint can generate only non-functional requirements.

VORD uses a simple graphical notation to represent a viewpoint (see Fig. 5). A rectangular box represents the viewpoint. The viewpoint identifier is shown on the top left-hand corner of the box and the viewpoint label in the lower half of the box. The viewpoint type is shown on the top right half of the box. A vertical line dropping from the left side of the box shows viewpoint attributes

Identifier	Denotes a unique viewpoint reference number
Label	Denotes a unique viewpoint name
Description	Describes the role of the viewpoint in the problem domain
Type	Traces the viewpoint to its parent class
Attributes	Characterise the viewpoint in the problem domain. Viewpoint attributes represent the control information provided by the viewpoint to the system
Requirements	Denotes a set of requirements generated by viewpoint
Goals	Denote a set of abstract requirements generated by viewpoint. Goals are decomposed into specific requirements
Event scenarios	Describe the interaction between the viewpoint and the system
Specialisations	Denote viewpoint subclasses
History	Describes the evolution of viewpoint components

Fig. 2. Viewpoint template.

Identifier	Denotes unique requirement reference number
Statement	The description in natural language, of the requirement
Rationale	Denotes the reason for the requirement
Type	Denotes the requirement type (functional, non-functional or subtype)
Source	Denotes the requirement source or sources
Priority	Denotes the priority of the requirement
Specification	Denotes the specification of the requirement in an appropriate notation

Fig. 3. Requirement template.

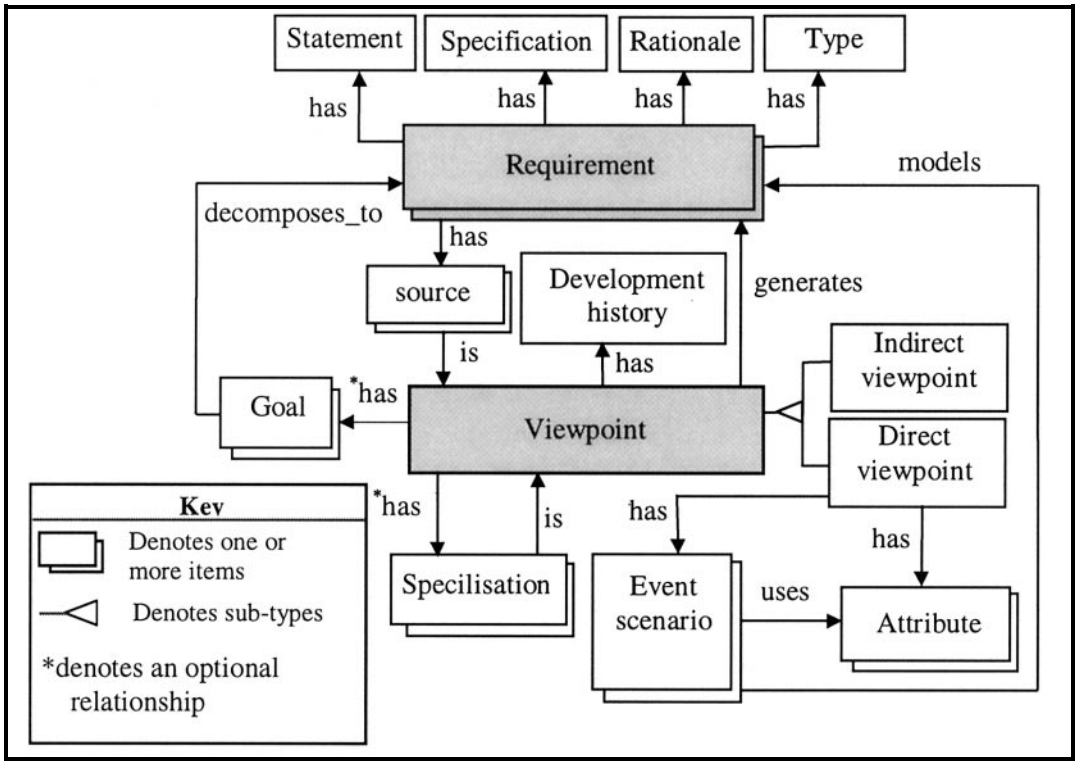

Fig. 4. Viewpoint information structure.

(discussed in section 3.2). A viewpoint may also have one or more specialised types (shown to the right).

The notation is accompanied by set of structured forms to collect viewpoint requirements and other related information. Figure 6 shows the VORD process model. The viewpoints and their accompanying documentation are products of the process, and are used as inputs to the requirements review process [14].

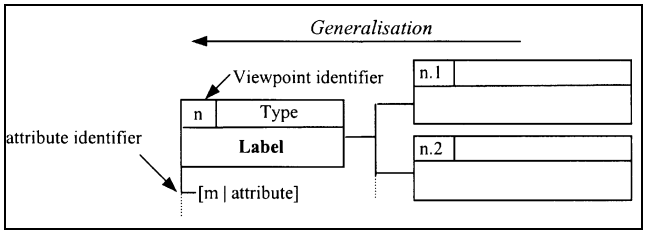

Fig. 5. Viewpoint notation.

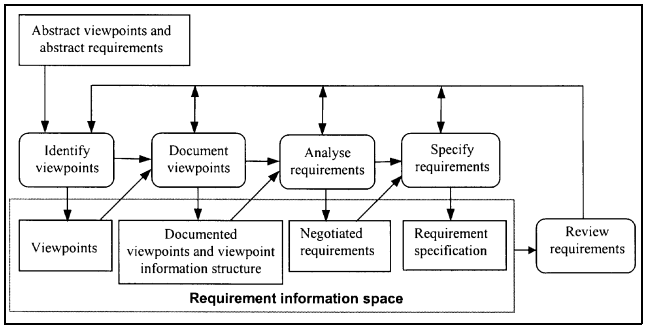

Fig. 6. VORD process model.

3.1. Identifying Viewpoints

We have identified a number of abstract viewpoints that act as the starting point for identifying application specific viewpoints. Figure 7 shows a set of the abstract viewpoint classes. This is a greatly reduced set of

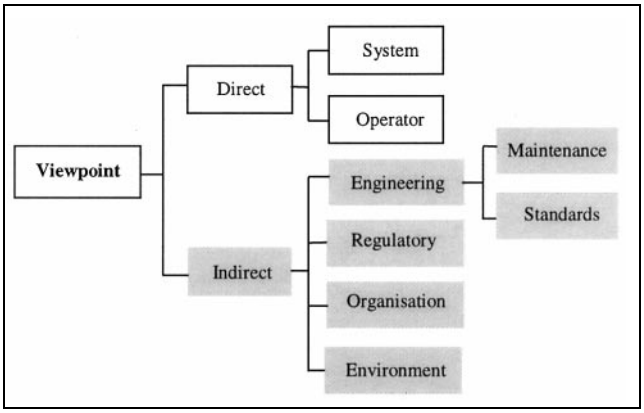

Fig. 7. Abstract viewpoint classes.

viewpoints, and organisations may identify abstract viewpoints that are more relevant to their applications domains.

The method of viewpoint identification involves a number of stages:

1. Prune the viewpoint class hierarchy to eliminate viewpoint classes, which are not relevant to the specific system being specified.
2. Consider the system stakeholders, i.e. those people who will be affected by the introduction of the system. If these stakeholders fall into classes, which are not part of the abstract class hierarchy add these classes to it.
3. Using a model of the system architecture, identify sub-system viewpoints. This model may either be derived from the existing system models or may have to be developed as part of an RE process. Sub-system viewpoints are instances of the system viewpoint, and include all systems that are going to interface with proposed system.
4. Identify system operators who use the system on a regular basis, who use the system on an occasional basis and who request others to use the system for them. All of these are potential viewpoints.

For each indirect viewpoint class which has been identified, consider the roles of the principal individual who might be associated with a class. For example, under the viewpoint class 'customer', we might be interested in the roles of 'regulations officer', 'maintenance manager', 'operations manager' etc. There are often viewpoints associated with these roles.

Based on this approach, the structure of the viewpoints identified for EDDIS is shown in Fig. 8. Direct viewpoints are shown in clear rectangles and the indirect viewpoints in grey. The EDDIS user viewpoint is a

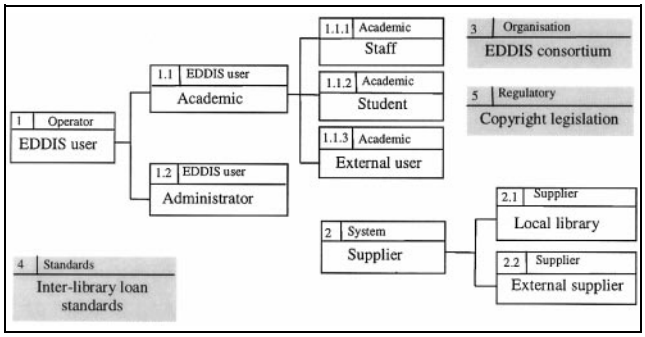

Fig. 8. Viewpoint structure for EDDIS.

direct viewpoint and comprises the 'academic' and 'administrator' viewpoints. The rationale for subclassing the EDDIS user viewpoint as shown is twofold:

- The viewpoints identify the main roles and responsibilities of the EDDIS user.
- The viewpoints allow us to be quite specific about the requirements of the users.

The general EDDIS user, for example, requires search and locate services from the system but the lower viewpoints require progressively specialised services. Structuring viewpoints in this manner also allows VORD to accommodate similar services with differing qualities as a result of the constraints imposed on them.

All identified viewpoints are accompanied by a brief description explaining their role in the problem domain (Fig. 9).

3.2. Documenting Viewpoints

3.2.1. Viewpoint Attributes

Viewpoint attributes represent values that characterise the viewpoint in the problem domain. They are things

Identifier	Label	Description
1	EDDIS user	Represents the general EDDIS user. These include people who use EDDIS for academic purposes and those who are concerned with the system administration
1.1	Academic	Represents people using EDDIS for academic purposes. Academic users include staff, students and users from outside of the University (external users)
1.2	Administrator	Represents the person concerned with the administration of EDDIS. The main role of the EDDIS administrator is to maintain user registration, collect system statistics for management purposes and to ensure correct system operation
2	Supplier	Represents the agents (systems) responsible for supplying documents to EDDIS. Document suppliers may be the local and/or external agencies
3	EDDIS consortium	Represents the 4 universities responsible for commissioning EDDIS
4	Inter-library loan standards	Represents the inter-library loan standards associated with the retrieval, format and delivery of electronic documents
5	Copyright legislation	Represents UK and international copyright legislation

Fig. 9. Viewpoint description.

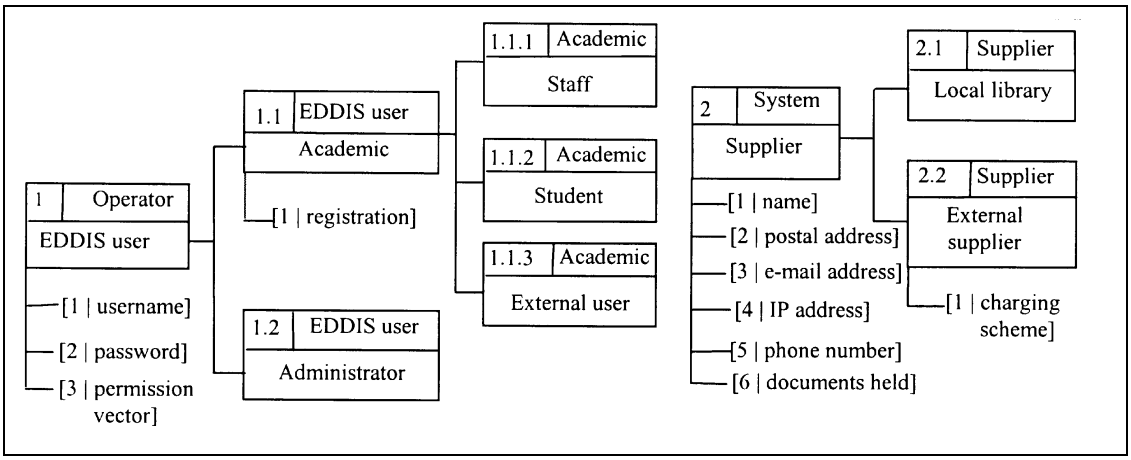

Fig. 10. EDDIS viewpoints with attributes.

'contained' or 'owned' by the viewpoint, for example, a customer viewpoint in a banking environment may have a *name* and an *account* as attributes. Attributes are important because they represent data that is consumed by the system operations. In VORD, attributes are documented for direct viewpoints because they represent control information and appear as parameters in event scenarios (described in Section 3.4).

Viewpoint attributes are also generalised from right to left (Fig. 10). All EDDIS user viewpoints have a username and a password to facilitate access to EDDIS. In addition all EDDIS users have a permission vector that holds a set of database ids, catalogue ids, services ids and accounts accessible to the user. All the academic viewpoints have a registration and the EDDIS administrator viewpoint has an administration password. The supplier viewpoint has a name, a postal, e-mail and IP address, phone number and the set of documents held. External suppliers may have a charging scheme for the documents supplied. A specific attribute is identified by the viewpoint identifier (in bold) and followed by the attribute number. Thus, for example, the 'registration' attribute appearing in the academic viewpoint has the identifier **1.1**.1.

Apart from an identifier and label, it is also important to define the structure of a viewpoint attribute. Many attributes are not atomic structures, and may have several parts. The EDDIS user password attribute, for example, needs to have a component that allows for the differentiation of the general user and EDDIS administrator. Structural definition can be done incrementally at varying levels of abstraction as the specification develops, and more information becomes available. VORD uses the BackusNaur Form (BNF) notation for this purpose. Thus, at a high level, we can define the permission vector attribute as shown in Fig. 11:

<permission_vector> ::= <$P_{databases}$> <$P_{catalogues}$> <$P_{service}$> <P_{store}> <$P_{printers}$> <$P_{parameters}$> <$P_{suppliers}$>

where:

$P_{databases}$	denote a set of user permitted databases
$P_{catalogues}$	denote a set of user permitted catalogues
$P_{service}$	denote a set of user permitted services
P_{stores}	denote a set of user permitted storage locations
$P_{printers}$	denote a set of user permitted printers
$P_{parameters}$	denote a set of user alterable environment parameters
$P_{suppliers}$	denote a set of user permitted suppliers

Fig. 11. User permission vector.

3.2.2. EDDIS User Requirements

The EDDIS user requirements are shown in Fig. 12. The requirements are derived from the natural language requirements in Section 2.1 and structured around viewpoints. This is an iterative process that required continuous verification with the user requirements document. The requirements have been numbered to reflect their viewpoint association. The requirement identifier comprises the viewpoint identifier (bold part), followed by the requirement number. A requirement that extends another is indicated extending the numbering of the parent requirement. We have retained the natural language statements shown in Section 2.1 for traceability reasons.

The documentation shows the viewpoint identifier (Id) and the requirement details. The academic viewpoint has no specific requirements of its own but inherits the requirements of the EDDIS user viewpoint. Similarly, the student and external user viewpoints inherit the requirements of the academic viewpoint. In addition to inheriting the requirements of their parent classes, viewpoint specialisations may also have specific require-

Viewpoint		Requirement		
Id	**Label**	**Id**	**Description**	**Source**
1	EDDIS user	**1**.1	*System access* The System will be accessed using a valid username and password	EDDIS con.
		1.2	*Document search* The system will allow a user to search a set of permitted databases for documents	EDDIS con.
		1.2.1	*Search criteria* The document search function will support variable search criteria, including: author, title, ISBN & keyword	EDDIS con.
		1.3	*Document locate* The system will allow a user to search a set of permitted catalogues to determine the location of documents	EDDIS con
		1.4	*Document read* The system will allow a user to read, print and store documents to specified printers and locations	EDDIS con.
		1.4.1	*Printing* The system will allow printing only to authorised printers, set in the user permission vector	EDDIS con.
		1.4.2	*Storage* The data stored by the use will be determined by account, and set by the system administrator	EDDIS con.
		1.4.3	*Output format* The output will indicate, in sequence: total number of items found, the number already in the output set, and the number of new items	EDDIS con.
		1.4.4	*Output result* The output results will be kept for the duration of the user session	EDDIS con.
1.1	Academic			
1.1.1	Staff	**1.1.1**.1	*Document order* The system will allow a user to order documents from specified suppliers	EDDIS con.
		1.1.1.1.1	*Unique identifiers* All document orders will have unique identifiers	EDDIS con.
		1.1.1.1.2	*Copyright enforcement* The system will request a user to sign a copyright declaration form for all document orders	Copyright legislation
		1.1.1.1.3	*Supplier terms* Supplier terms will accompany all document orders	EDDIS con.
		1.1.1.1.4	*Status enquiry* The system will allow a user to query the status of ordered documents	EDDIS con.
1.1.2	Student			
1.1.3	External user			
1.2	EDDIS administrator	**1.2**.1	The system will allow a user to register users	EDDIS con.
		1.2.1.1	The system will allow a user to remove users	EDDIS con.
		1.2.2.1	The system will allow a user to modify registration information	EDDIS con. EDDIS con.
		1.2.2	The system will allow a user view usage information	EDDIS con.
		1.2.2.1	The report format for EDDIS usage will be organised as follows, over defined periods of time (i.e. daily, by date, weekly, monthly): 1. document searches executed 2. document locations executed 3. documents supplied 4. pages supplied 5. document orders not supplied 6. document orders cancelled and ordered by: use, cost centre and institutional level	EDDIS con.
		1.2.3	The system will allow a user to access restricted database information	EDDIS con.
		1.2.4	The system will allow a user to access restricted catalogue information	EDDIS con.
		1.2.5	*Non-digitised document* The system will allow a user order non-digitised documents	EDDIS con.

Fig. 12. Requirements for EDDIS user.

ments. The management services, for example, are provided only to the EDDIS administrator.

It is important that each requirement has an accompanying rationale. Requirement rationale is an important tool for gauging the relevance and level of importance of a requirement, particularly when trade-offs need to be made between requirements. In VORD, a requirement rationale is represented as a natural language text that accompanies each requirement. We do not have the space here to show the rationale for EDDIS requirements.

3.2.3. Document Supplier Requirements

The document supplier receives requests for documents from the users through the 'document order' service. Documents sent by the supplier are received by EDDIS and passed onto the user account. Because it is a system in its own right, the document supplier may provide services as well as generate requirements; we therefore need to analyse it from two perspectives:

- As a direct viewpoint to EDDIS. We need to establish what services, if any, the document supplier expects from EDDIS, and what constraints it imposes on EDDIS.
- As a system with EDDIS as a direct viewpoint, and the indirect viewpoints acting as potential sources for constraints. We need to establish what services EDDIS requires from the document supplier and what constraints it imposes on the document supplier. We also need to establish how it is affected by the indirect viewpoints.

Document supplier as viewpoint. The requirements generated by the document supplier are shown in Fig. 13.

Document supplier as a system. The requirements generated by EDDIS are shown in Fig. 14. No requirements are generated with respect to the indirect viewpoints in this case.

The kind of analysis done for the document supplier is critical for distributed systems. The supplier requirements are modelled as part of the event scenario for the document order service (see Fig. 20).

3.2.4. Requirements for Indirect Viewpoints

EDDIS has three indirect viewpoints: EDDIS consortium, document standards and copyright legislation. The EDDIS consortium viewpoint is concerned with issues affecting the system development and the quality of services to be provided. Most of the constraints on the EDDIS user services originate from this viewpoint. The document standards viewpoint is concerned with document interface and ordering standards. The copyright legislation viewpoint is concerned with the copyright requirements associated with the documents (Fig. 15).

Viewpoint		Requirement		
Id	**Label**	**Id**	**Description**	**Source**
2	Supplier	**2**.1	The system will retrieve information using the information retrieval standard Z39.50	Inter-library loan standards
		2.2	All system requests for documents will comply with the ISO 10160-1 format for Interlibrary Loan Application Protocol	Inter-library loan standards
		2.3	The system will make a provision for e-mail communication between the supplier and the person making an order	Supplier

Fig. 13. Document supplier requirements.

Viewpoint		Requirement		
Id	**Label**	**Id**	**Description**	**Source**
6	EDDIS	**6**.1	The system will supply documents to EDDIS on request	EDDIS con.
		6.2	Supplied documents will be accompanied by supplier terms	EDDIS con.
		6.3	The system will, on request, supply status information on ordered documents	EDDIS con.

Fig. 14. EDDIS requirements.

Viewpoint		Requirement		
Id	Label	Id	Description	Source
3	EDDIS consortium	3.1	The system development will be on UNIX platform	EDDIS con.
		3.2	The system development language will be C++	EDDIS con.
		3.3	A proof of concepts system will be delivered by 1/3/97	EDDIS con.
		3.4	The system will have a built-in mechanism for graceful degradation of services in the event of system failure	EDDIS con.
		3.5	The document location codes will be based on the current edition of British Library Document Supply, Directory of Library Codes	EDDIS con.
		3.6	The management services will be available 99% of the time	EDDIS con.
		3.7	EDDIS will maintain a reasonable quality of service to its users	EDDIS con.

Fig. 15. Requirements for indirect viewpoints.

3.3. Identifying Constraints

Because they are constraints on system services, non-functional requirements greatly influence the design solution [15]. In real-time systems, performance requirements may be of critical importance, and functional requirements may need to be sacrificed in order to achieve minimally acceptable performance. The expression of non-functional requirements poses several problems:

- Certain constraints, for example response time to failure, are related to the design solution that is unknown at the requirements stage.
- Other constraints, especially those associated with human engineering issues, are highly subjective and can only be determined through complex empirical evaluations.
- Non-functional requirements tend to be related to one or more functional requirements. Expressing functional and non-functional requirements separately obscures the correspondence between them, whereas stating them together makes it difficult to separate the functional and non-functional considerations.
- Non-functional requirements tend to conflict and contradict each other. The process of arriving at a trade-off in these conflicts depends on the level of importance attached to the requirement and the consequence of the change on other requirements and the wider system goals.

A number of frameworks for representing and using non-functional requirements have been proposed. Mylopoulos [16] proposes a goal-driven framework which provides for the representation of non-functional requirements in terms of interrelated goals. The goals are refined through refinement methods and can be evaluated in order to determine the degree to which a set of non-functional requirements is supported by a particular design. Chung [17] describes an incremental approach that shows how a historical record of the treatment of non-functional requirements during the development process can also serve to systematically support evolution of the software system. In the approach, changes are treated in terms of adding or modifying non-functional requirements, or changing their importance, and changes in design decisions or design rationale.

Constraint			Constrained
Id	Description	Type	Requirement
3.1	The system development will be on UNIX platform	Implementation	All
3.2	The system development language will be C++	Implementation	All
3.3	A proof of concepts system will be delivered by 1/3/97	Delivery	All services
3.4	The system will have a built-in mechanism for graceful degradation of services in the event of system failure	Reliability	All services
3.5	The document location codes will be based on the current edition of British Library Document Supply, Directory of Library Codes	Standard	1.3
3.6	The management services will be available 99% of the time	Reliability	1.2.1, 1.2.2, 1.2.3, 1.2.4, 1.2.5
3.7	The system will maintain a reasonable quality of service to its users	Reliability	All services
2.1	The system will retrieve information using the information retrieval standard Z39.50	Standard	1.2, 1.3, 1.4 1.1.1.1
2.2	All system requests for documents will comply with the ISO 10160-1 format for Interlibrary Loan Application Protocol	Standard	

Fig. 16. Constraints on EDDIS services.

In VORD, non-functional requirements are viewed as constraints on viewpoint services, control information or the system in general. Viewpoints may privately impose constraints on received services and different viewpoints may place different constraints on similar services, as is the case with performance constraints on some real-time systems. Viewpoints may also generate non-functional requirements that constraint the services provided to other viewpoints, as is the case with indirect viewpoints.

3.3.1. Constraints on Services

This section identifies and documents the constraints affecting the services provided by EDDIS. Figure 16 shows the constraints on the EDDIS user services and the affected services. Constraints can be traced to non-functional requirements through their identifiers.

3.4. Describing System Behaviour

VORD uses event scenarios to model system behaviour. Event scenarios describe how the system interacts with its environment. They capture the control relationships between the proposed system and its environment. The provision of a viewpoint service is the culmination of a series of events arising from the viewpoint, and filtering through levels of control to entities within the system that are ultimately responsible for its provision. Event scenarios provide a modular and traceable way of modelling system behaviour.

An event scenario is defined as a sequence of events together with exceptions that may arise during the interchange of information between a direct viewpoint and the system. A normal sequence of events may have exceptions at various points in the event sequence. At the system level, exceptions cause a transfer of control to exception handlers. VORD uses an extended state transition model based on the model proposed by Rumbaugh [18]. Exceptions are shown in grey arrows and normal sequences in black. A transition is triggered by an event and/or preconditions, which must be satisfied before the transition can take place. An event may include an optional set of parameters, and may be accompanied by a set of actions. The level of abstraction used in event scenarios may be varied to improve their readability. We use semi-formal descriptions in the examples shown here. The following key is to be used when reading event scenarios:

1. *event(optional set of parameters)* (event labels are shown italicised)
2. [precondition] (preconditions are enclosed in square brackets)
3. /actions (action labels start with a forward slash)

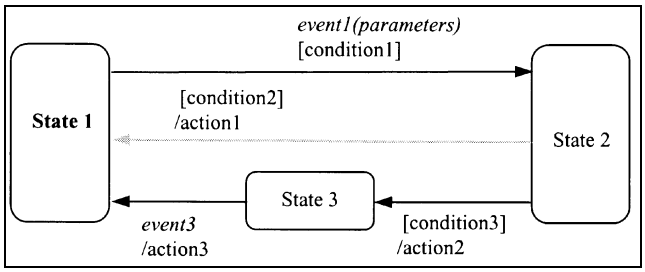

Fig. 17. Notation for extended state transition model.

Figure 17 shows the state transition notation used. The normal sequence of events is shown in black and exceptions in grey. The initial state is labelled in bold typeface.

3.4.1. Event Scenario for System Access

The 'system access' service is concerned with providing the user with access to EDDIS services and setting up the default user environment. Figure 18 shows the event scenario for the system access service. The system is initially in *idle* state until the user logs on with a valid username and password. A valid username and password causes the system to go into a *ready* state where it opens a session and sets up the environment based on the user *permission vector*. An invalid username or password causes an exception to be raised.

When the system is in the *ready* state, the user can select the desired service; this is equivalent to sending the system a *select(service)* event. To maintain a reasonable quality of service, all EDDIS services are constrained by the level of demand. A system monitor keeps track of service requests and allows or disallows them accordingly. When the user selects a service, the system goes into the *service* state, where it displays a

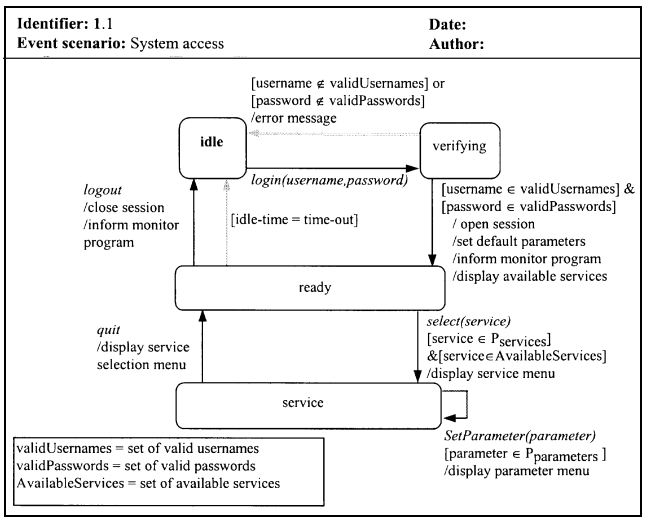

Fig. 18. Event scenario for system access.

menu for the selected service. A 'time-out' exception will occur if the user fails to make a choice within a permitted time, referred to as 'idle-time'. A *quit* event in the *service* state causes a transition to the *ready* state.

The *ready* state forms the common point for accessing all other EDDIS services. A user is allowed to access only those services set in the user's permission vector. The service state refers to the system in a specific *service* state, for example, document search.

EDDIS provides two types of user-accounts: individual and group. Users of individual accounts are allowed to alter the environment of the selected service, for example, to select the initial set of databases and catalogues to be searched. The *setParameter(parameter)* event allows users with individual accounts to do this. Users with group accounts are not permitted to do this. We will limit our illustration here to the scenarios associated with the document search and document order services.

3.4.2. Event Scenario for Search Service

The document search service allows the user to search a set of databases, D, to obtain details of documents that match a key based on the search criteria. The search criteria may be the author's name, document title, document ISBN or a keyword. The databases searched are determined by the set of user permitted databases ($P_{databases}$). The event scenario for the search service is shown in Fig. 19. The result from the search is placed in a 'search basket' and displayed as a set of document_ids.

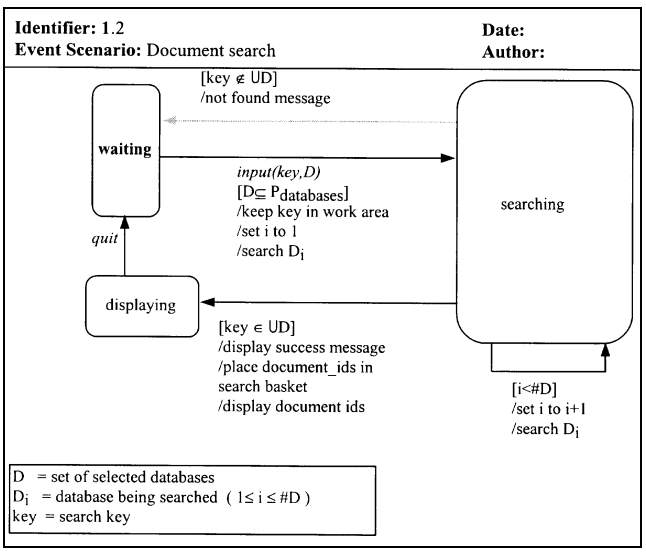

Fig. 19. Event scenario for document search.

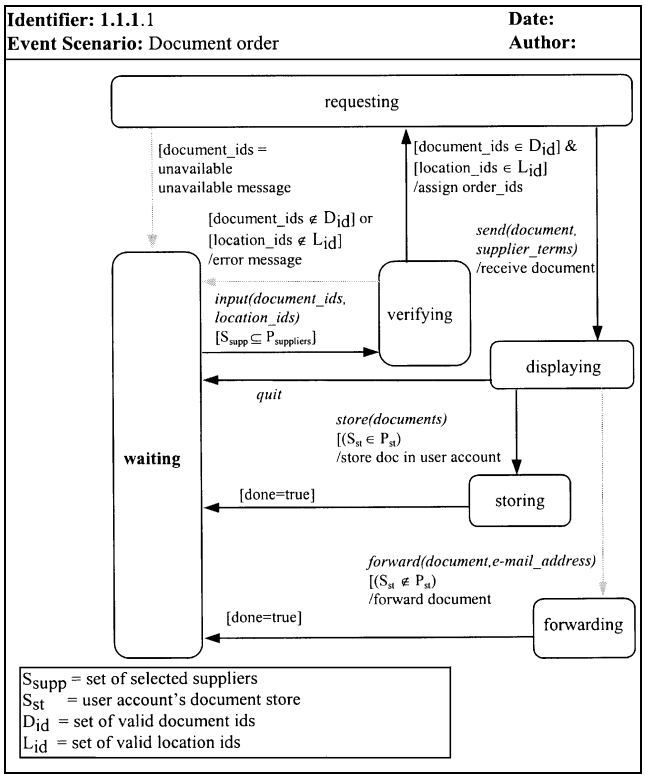

Fig. 20. Event scenario for order service.

3.4.3. Event Scenario for Order Service

The document order service allows users to order documents from a list of permitted suppliers. The output from this service is initially a set of order-ids, and later documents.

Figure 20 shows the event scenario for the document order service. The set of user permitted suppliers represented by $P_{suppliers}$. The service is initiated by an *input(document_id,location_id)* event. The documents are stored locations determined by the user's permission vector. If a request is made for a non-digitised document, a supplier confirms availability via e-mail to the user, and sends the document to the Librarian.

3.5. Analysing Requirements

The objective of requirements analysis is to establish that viewpoint requirements are correct, 'complete' and feasible. There are four main stages to this analysis.

1. Correctness of viewpoint documentation: viewpoint documentation is checked to ensure consistency and completeness.
2. Conflict analysis: conflicting requirements are exposed and ways to resolve the conflicts proposed.

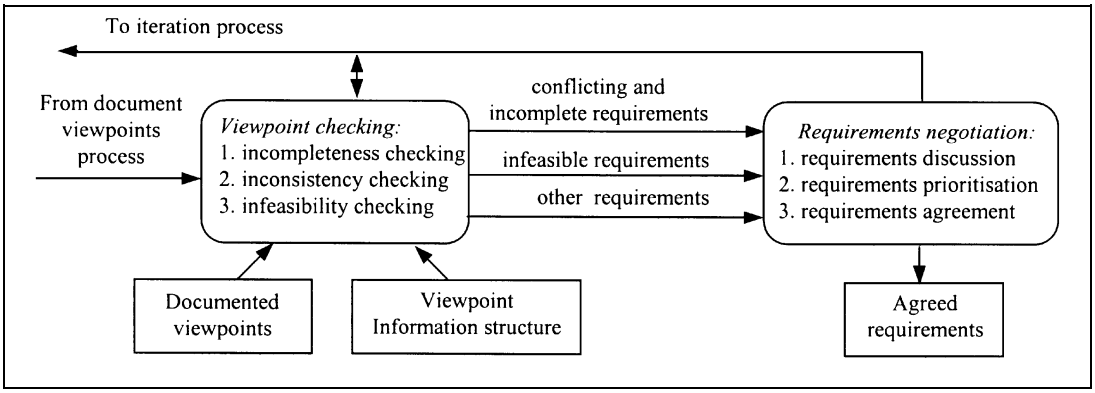

Fig. 21. VORD requirements analysis process.

3. Feasibility analysis: requirements are checked for feasibility with respect to the development schedule and available resources.
4. Changes are negotiated for 'problem' requirements and agreed.

Figure 21 shows the VORD requirements analysis process.

3.5.1. Completeness Checking

Completeness involves verifying that a viewpoint has been correctly and completely documented. In Section 3, we defined a viewpoint information template as comprising an identifier, a description, a type, attributes, requirements, goals, event scenarios, specialisations and history. Although some of this information must appear on all viewpoints, other information may be omitted depending on whether the viewpoint is direct or indirect. Figure 22 shows the relationship between a viewpoint type and the need for corresponding documentation.

This scheme is used to verify the completeness of viewpoint documentation and is explained as follows:

Information \ Viewpoint type	Direct	Indirect
Identifier	yes	yes
Label	yes	yes
Description	yes	yes
Type	yes	yes
Attributes (Control information)	yes	no
History	yes	yes
Service	yes*	no
Goal	optional	optional
Non-functional requirements	optional	yes
Specialisations	optional	optional

Fig. 22. Checklist for viewpoint documentation.

1. A 'yes' means that the documentation must be present in the viewpoint; for example, a viewpoint must be uniquely labelled and traceable to abstract viewpoints.
2. A 'no' means that the corresponding documentation is not part of the viewpoint; for example, an indirect viewpoint does not receive services or have attributes.
3. An 'optional' means that the documentation may optionally be present in the viewpoint. For example, viewpoints may or may not have specialisations; and direct viewpoints may or may not have non-functional requirements. Where an optional documentation is present, it must be checked against other related documentation (see viewpoint information structure in Fig. 4).
4. 'yes*' denotes a set of information, at least one of which must be documented in the viewpoint. In other words the set refers to information that may be present in whole or part, in the viewpoint. For example, some direct viewpoints receive services and provide control information, others provide only control information.

Correctness checking is also intended to ensure that there is continuity between individual event scenarios and that control information used in the event scenarios can be traced to viewpoints (Fig. 23).

3.5.2. Conflict Checking

Conflicting requirements may arise from contradictions among individual viewpoints. Viewpoints have differing stakes and interactions with the intended system and have requirements that are closely aligned with these interests. Non-functional requirements tend to conflict and interact with other system requirements. This kind of conflict may be quite specific, as in the following two cases:

Service	Event	Event source	Proposed action
System access	login(username, password)	EDDIS user	None
	logout	EDDIS user	None
	select(service)	EDDIS user	None
	setParameter(parameter)	EDDIS user	None
	quit	EDDIS user	None
Document search	input(key,D)	EDDIS user	None
	quit	EDDIS user	None
Document order	input(document_ids, location_ids)	Staff	None
	send(document, supplier_terms)	Supplier	None
	store(documents)	Staff	None
	forward(document,	Staff	None
	e-mail_address)	Staff	None
	quit		

Fig. 23. Partial scenario information analysis.

Requirement 1 : 1.2 Document search

Requirement 2	Conflict description	Proposed action: Requirement 1	Proposed action: Requirement 2
1.1 System access	None	None	None
1.2.1 Search criteria	None	None	None
3.7 Quality of service	Not clear how this going to affect the document search service	None	Rewrite to clarify the meaning of 'reasonable quality of service'

Fig. 24. Partial conflict analysis matrix.

- Where the provision of a service across viewpoints is associated with different constraints of the same general type.
- Where the provision of a service across viewpoints is associated with similar constraint types; but differing constraint values.

*It may also be the case that a requirement in one viewpoint contradicts a requirement in another viewpoint. These types of conflicts can be exposed by analysing the constraints associated with a particular service, for consistency, and by analysing a viewpoint requirements against the requirements in other viewpoints for contradictions. In addition to these specific viewpoint requirements, there are high-level organisational and other global requirements that define the general quality attributes of the intended system. Quality goals are normally generated by indirect viewpoints that make up the organisation purchasing the software system. Requirements that are affected by these quality attributes must be analysed against the quality attributes for consistency.

Individual measures of quality may also conflict with each other, and compromises may have to be reached. The solution sought is an optimum balance of factors rather than an ideal solution. The checking model adopted by VORD is based on ensuring that information can be presented in a way that manual analysis is simplified. Figure 24 shows part of conflict analysis matrix for the document search requirement.

3.5.3. Feasibility Checking

Feasibility checks are intended to ensure that requirements are feasible in the context of the resources and schedule available to the system development. Infeasible requirements may be delayed, deferred or removed (see Fig. 25).

3.5.4. Requirements Negotiation

The list of incomplete, conflicting and infeasible requirements together with proposed actions is taken through a process of requirements negotiation. The

Requirement		Suggested changes	Reason	Agreed action
Id	Label			
1.1.1.2	Document order	Document order event scenario be modified to include the copyright requirement	A signed copyright acceptance form must accompany all document orders. This information is missing from the event scenario	The event shall be modified thus: *input(document_ids, location_ids)* modified to read: *input(document_ids, location_ids, ©)* where © is a signed copyright acceptance form
3.7	EDDIS must maintain a reasonable quality of service to users	Rewrite requirement to clarify what is meant by 'reasonable quality of service'	It is not clear what is meant by 'reasonable quality of service'	EDDIS must be able to automatically monitor and control the number of people logged onto the system at any time to maintain a reasonable quality of service to its users. The program will monitor the users logged on against a preset maximum and prevent additional users logging on when the preset a maximum is reached
1.2.5	Non-digitised receipts	Requirement to be deferred to next release of EDDIS	It is not possible to incorporate this requirement in the first release of EDDIS given the schedule	The requirement is deferred to the next release of EDDIS
3.3	Proof of concept system by 1/3/97	Proof of concept system to be delayed to 1/12/97	It is not possible to meet this delivery deadline due to the restrictive schedule	The proof of concept date has been put forward to 1/12/97

Fig. 25. Summary of analysis and changes.

Item	Checklist question
Viewpoint	1. Does the viewpoint have a unique identifier? 2. Is the viewpoint described clearly? 3. Have all viewpoint requirements been identified? 4. Have all specialisations been identified for viewpoint? 5. If viewpoint is direct, have all viewpoint event scenarios been described? 6. Are all viewpoint attributes traceable to event scenarios? 7. If viewpoint is system, has separate viewpoint documentation been done?
Requirement	1. Does the requirement have a unique identifier? 2. Is requirement described clearly? 3. Is requirement traceable to a viewpoint? 4. Have all the dependent requirements been identified? 5. Are all requirement sources traceable to viewpoints? 6. If requirement is functional? Has an event scenario been described for requirement?
Event scenario	1. Are all events and parameters traceable to viewpoints? 2. Does scenario describe requirement adequately? 3. Have all exception scenarios been identified?

Fig. 26. Viewpoint checklist.

requirements negotiation is mainly a human process that is supported by the information collected during requirements analysis. It is important to note that a requirement may conflict with several other requirements. This may a result in the requirement having several proposed changes. It is important to ensure that the suggestions do not in themselves conflict. Figure 25 shows a summary of some of the problems encountered during the analysis of EDDIS requirements, and the agreed actions.

3.6. Requirements Review

The documents produced during the VORD requirements process are not only intended as inputs to the next stage of the process, but also form inputs to the review process. The review process is intended to check the documents for consistency, traceability, completeness, understandability and conformance, among other things. VORD provides a list of checklist questions to help reviewers with this process. The complete list is shown in Fig. 26.

REQUIREMENTS DOCUMENT
SPECIFIC REQUIREMENTS SECTION

SPECIFIC REQUIREMENTS
Viewpoint
1. EDDIS User

A Description
// description of viewpoint
Represents the general EDDIS user. These include people who use EDDIS for academic purposes and those who are concerned with the system administration.

B Type
//this section defines the viewpoint
/Direct/Operator

C Specialisations
//this section provides a list of viewpoint specialisations
1.1 Academic
1.2 EDDIS Administrator

D Requirements
//viewpoint requirements are described in this section

D1 Services
//viewpoint services or functional requirements are described here

1.1 System access
Statement:
The system shall be accessed using a valid username and password...
Source: 3 (EDDIS consortium)
Priority: High <Can be high, medium, low based on importance, resources, and risk>
Rationale: A basic system service, all other services are dependent on it
Event scenario: E1.1
Specification: < Specification of service>

1.2 Document search
:

1.3 Document locate
:

1.4 Document read
:

D2 Non-functional Requirements
//Non-functional requirement associated with viewpoint goes here
//each non-functional requirement has a unique identifier, description, source,
//priority and a list of affected requirements

E History
//References to the development history of viewpoint and its components goes here

Fig. 27. Part of EDDIS requirements specification document.

3.7. Requirements Specification

For most non-trivial systems, there is a need to translate the result of the requirements analysis process into a standard requirements document, to conform to industrially recommended practices for specifying software. This section describes how the viewpoint-oriented requirements for EDDIS can be translated into a standard specification document. The requirements document standard used is adapted from the recommendations of the IEEE standard 830-1993 [19]. Figure 27 shows part of the EDDIS requirements document. The VORD requirements document is structured in the context of viewpoints to maintain traceability with viewpoints. Section A provides a short description of the viewpoint and section B the viewpoint type. Section C lists all viewpoint specialisations in terms of their references. Section D provides the development history of the viewpoint and its components). Section E describes the viewpoint requirements. In addition, services have an event scenario and specification. The description of non-functional requirements includes references to affected services.

4. Lessons Learnt

The process of specifying EDDIS requirements with VORD has been largely successful. A proof of the concept system is planned for operation at the four development sites by December 1998. However, we have noted a number of general weaknesses with the method. These included:

- *Lack of rapid change management techniques.* In VORD, as in most requirements methods, the process of change management is compounded by the need to trace and analyse large amounts of complex inter-related information. Many requirements methods address this problem by freezing requirements at fixed points within the life cycle; however, this may lead to systems that fail to meet the real business needs of the system procurer [15].

  There is a need in requirements engineering for a rapid and cost-effective means of addressing this problem. One way to address the problem would be to look at ways of managing change through rapid visualisation of change scenarios. Scenario-based techniques have previously been used in requirements engineering for supporting early requirements validation and providing guidelines to build prototypes. However, these are largely concerned with requirement elicitation and are not designed for analysing and assessing the impact of change. The author is currently investigating use of scenario-based visualisation techniques as a means of analysing, tracing and controlling requirements change from early requirement formulation through to later system development. The intention is to be able to model both static change scenarios showing static dependencies and dynamic change scenarios that show how control ripples through the system.
- *Lack of support for the social process.* It is important to understand that requirements engineering is both a technological and a complex social process. Conventional technically oriented approaches to requirements elicitation only address part of the problem. In VORD, viewpoint and early requirements identification process might benefit from the support of an approach that takes into account the social aspects of system development. One such approach is the USTM method developed by Macaulay [20]. USTM is a cooperative requirements capture method in which the social process is managed through the use of a human facilitator. USTM is organised around workshops and workgroups where stakeholders interact and brainstorm in identifying objects and tasks in the problem domain. The role of the facilitator is to maintain the group focus and cohesiveness by managing the agenda, observing the group process, diagnosing problems, design solutions and making appropriate interventions. Consolidation workshops are used to assess the quality of the information gathered and to reassess the business case for the system, and to plan for the next phase of work. A different, but potentially useful approach is proposed by Potts [21]. The approach, based on an inquiry model, is intended to support the requirements identification process by continuously refining vague stakeholder requirements until they are sufficiently acceptable. The model uses a hypertext model where each requirement is a separate node in the hypertext. The process works by getting stakeholders to challenge the proposed requirements by attaching annotations. The requirements evolve through a process of change requests, discussion and modification.
- *Lack of support for collaboration.* Currently VORD does not support user/engineer collaboration; however, it is possible to extend it to provide such support. This is quite important because the process of developing large software systems involves the participation of experts at various levels of the software development and application area. It is important that the requirements method provides support for the roles, interaction and responsibilities of the various participants. It would be desirable if such a framework supported not only the need for communication amongst the participants but also the requirements process. This would facilitate remote working by participants with a subsequent reduced need for face-to-face meetings. In the case of EDDIS, participants felt the need to communicate outside scheduled meetings largely to seek clarification or opinion on assigned actions.

  A starting point might be to use VORD to formulate the requirements of the collaborative framework. In such a scenario, viewpoints would represent the various stakeholders (participants). Services required by the stakeholders would represent the additional functionality expected of VORD. Group interaction could be modelled as event scenarios. An interesting aspect of this formulation would the process of resolving the constraints associated with group interaction.
- *Lack of guidelines.* Viewpoint-based requirements methods have been around for several years now; however, there is a general lack of information on their suitability for use on large systems. This particular project was largely successful, because VORD was primarily developed for specifying interactive systems, a class of systems which EDDIS closely fitted. While the service model used by VORD is quite intuitive, it may have difficulty addressing

problems associated with systems that do not fit neatly into the service-oriented systems (SOS) paradigm. Service-oriented systems can be viewed as service-providing enterprises; they employ systems composed of people, computer hardware and software, and other mechanisms to perform service actions in the customer environment [22].

Clearly there is a need for a set of comprehensive guidelines on the suitability of use of viewpoint approaches with large systems. It is possible that more case studies may need to be carried out before such guidelines can be provided.

- *Limited interoperability of VORD toolset.* VORD is based on an integral toolset that is intended to provide support from initial requirements formulation through to detailed specification. The toolset provides a number of useful facilities including: support for viewpoint creation and documentation, consistency and completeness checking, event scenario modelling, reviews and report generation. However, the toolset lacks facilities that can enable users to share or port the information to other software tools and methods. This facility would provide a means of integrating the complementary strengths of various approaches to study aspects of the problem domain that may not be amenable to a single method or technique.

We are currently evolving VORD to address these problems and will report our results in a later publication.

5. Summary

This paper has demonstrated the practical utility of a viewpoint-based requirements approach, VORD, on a medium-sized interactive system.

- VORD viewpoints focus on user issues and organisational concerns. We have shown how they can be used to elicit varied stakeholder requirements. In EDDIS, the stakeholders have included EDDIS users, administrators, document suppliers, EDDIS consortium, document standards and copyright legislation.
- VORD provides an open and traceable framework within which other approaches and techniques may be incorporated to complement the method. It is possible, for example, to derive use cases from event scenarios; and to associate them with services. It is also possible to specify services using a variety of notations to promote understandability.
- We have demonstrated the importance of incorporating indirect viewpoints in the requirements engineering process. In EDDIS, this has enabled us to explicitly address the concerns generated by the EDDIS consortium, document standards and copyright legislation.
- We have shown how the explicit identification of viewpoints with services in VORD has made it possible to create a framework where many aspects related to system requirements can be integrated. VORD provides a framework where viewpoints, services, non-functional requirements and event scenarios can be integrated.
- Event scenarios have been used in VORD to describe the behaviour of the EDDIS system. Event scenarios provide a modular and concise way of describing the complex interactions between the EDDIS viewpoints and system.
- VORD has no predefined notation for specifying services. In large complex systems more than one notation may be needed to represent and communicate the requirements adequately.
- VORD supports the translation of requirements to a standard requirements document such as the IEEE standard 830-1993. This is important for conformity with industrially recommended software development practices.

Finally, to assist other researchers in comparing their requirements engineering techniques, we have identified a set of 10 comparators drawn from the lessons learnt with EDDIS and our experience in the field:

- *Accommodation of stakeholder concerns.* To what extent does the approach accommodate varying stakeholder requirements and concerns? The requirements analysis process is greatly aided by the ability of a method to separate stakeholder concerns while maintaining the correspondence between them.
- *Integration of different notations.* To what extent does the approach support the integration of different notations? There is no single requirements notation that can adequately articulate all the requirements of a system.
- *Definition of system environment.* To what extent does the approach support the definition of the system's environment? The requirements model is incomplete unless the environment with which the system interacts is modelled. This should include a description of the interaction between the system and its environment. The modelling of exception scenarios is particularly important.
- *Integration of functional and non-functional requirements.* To what extent does the approach support the integration of functional and non-functional requirements? Non-functional requirements tend to be related to one or more functional requirements.

- *Requirements analysis.* To what extent does the approach support consistency, completeness and feasibility checking?
- *User collaboration.* To what extent does the approach support collaboration between the intended system users and the requirements experts during the requirements formulation?
- *Guidelines.* To what extent does the approach provide guidelines on its suitability for use on large systems?
- *Standardisation.* To what extent are the results of the requirements analysis, as produced by the approach, translatable to a standard requirements document?
- *Change management.* To what extent does the approach support requirements change? It must recognised that requirements are built gradually over long periods of time and continue to evolve throughout the component's life cycle. The requirements definition process used should be tolerant of temporary incompleteness and adapt to changes in the nature of the needs being satisfied by the component?
- *Requirements validation.* To what extent does the approach help with the process of requirements validation?
- *Tool support.* To what extent is the approach or method supported by tools?

References

1. Darke P, Shanks G. Stakeholder viewpoints in requirements definition: a framework for understanding viewpoint development approaches. Requirements Eng 1996;1(2):88–104
2. Kotonya G, Sommerville I. Requirements engineering with viewpoints. Software Eng J 1996;11(1):5–11
3. Special issue on viewpoints in requirements engineering. Software Eng J 1996;11(1)
4. Dubois E, Hagelstein J, Rifuat A. Formal requirements engineering with ERAE. Philips J Res 1988;43(3/4):393–414
5. Leite JCSP, Freeman PA. Requirements validation through viewpoint resolution. IEEE Trans Software Eng 1991; 17(12):1253–1269
6. Mullery G. CORE: a method for controlled requirements specification. In: Proceedings of 4th international conference on software engineering. Munich, 1979
7. Finkelstein A, Kramer J, Nuseibeh B, Finkelstein L, Goedicke M. Viewpoints: a framework for integrating multiple perspectives in systems development. Int J Software Eng Knowledge Eng 1992;2(1):31–57
8. Sommerville I, Sawyer P. Viewpoints: principles, problems and a practical approach to requirements engineering. Ann Software Eng 1997;3:101–129
9. Finkelstein A, Kramer J, Goedicke M. Viewpoint-oriented software development. In: Proceedings of 3rd international workshop on software engineering and applications, Toulouse, 1990
10. Rubenstein HB, Waters RC. The requirements apprentice: automated assistance for requirements acquisition. IEEE Trans Software Eng 1991;17(3):226–240
11. Dobson JE, Blyth JC, Chudge J, Strens R. The ORDIT approach to organisational requirements. In: Requirements engineering: Social and technical issues, Academic Press, London, 1994, pp 88–106
12. Boehm B, Egyed A, Kwan J, Madachy R. Developing multimedia applications with the WinWin spiral model. In: Proceedings of 6th European software engineering conference, Zurich, 1997
13. Dardenne A, Fickas S, van Lamsweerde A. Goal-directed concept acquisition in requirements elicitation. In: Proceedings of IWSSD, 1996
14. Kotonya G, Sommerville I. Requirements engineering: processes and techniques. Wiley, Chichester, 1998
15. Sommerville I. Software engineering, Addison-Wesley, Reading, MA, 1996
16. Mylopoulos J, Chung L, Nixon B. Representing and using non-functional requirements: a process-oriented approach. IEEE Trans Software Eng 1992;18(6):483–497
17. Chung L, Nixon BA, Yu E. Using non-functional requirements to systematically support change. In: Proceedings of 2nd international symposium on requirements engineering, York, 1995
18. Rumbaugh J, Blaha M, Premerlani W, Eddy F, Lorensen W. Object-oriented modelling and design, Prentice-Hall, Englewood Cliffs, NJ, 1988
19. Mazza C, Fairclough J, Melton B, De Pablo D, Scheffer A, Stevens R, Jones M, Alvisi G. Software engineering guides, Prentice-Hall, Englewood Cliffs, NJ, 1996
20. Macaulay L. Requirements capture as a cooperative activity. In: Proceedings of 1st international symposium on requirements engineering, San Diego, 1993
21. Potts C, Anton A, Takahasi K. Inquiry-based requirements analysis. IEEE Software 1994;11(2):21–32
22. Greenspan S, Feblowitz M. Requirements engineering using the SOS paradigm. In: IEEE international symposium on requirements engineering, San Diego, 1993