Sommerville 1992) or naming differences (Leite & Freeman 1991).

Each of our research directions, when viewed in isolation, has both strengths and weaknesses. The main advantage from our data base is its power for storing real requirement specifications for composite systems. Existing techniques such as viewpoint-oriented software engineering (Nuseibeh et al. 1993) focus on representation but not storage. However, the deductive data base axioms do not provide reasoning mechanisms of sufficient power to reason about complex viewpoints on requirements. Retrieval of problem abstractions provides more powerful reasoning mechanisms and uses domain knowledge to focus these mechanisms on the fundamental features of viewpoints. Indeed, domain knowledge is needed for intelligent assistance for requirements engineers. However, the principle disadvantage of this approach is its use of a constrained domain modelling language which necessitates translation of viewpoints from other representations. The third research direction, comparing viewpoints using similarity matching, also provides more powerful reasoning mechanisms but for comparison between viewpoints which are represented using structured notations, thus focusing on both representation and domain knowledge differences between viewpoints. Its domain-independent generalisations provided in the meta-model can be complemented by the domain modelling approach which detects more semantic differences between viewpoints. Therefore, the strength of the work described in this paper is the combination of the three research directions to maximise their individual advantages.

Furthermore, these directions will be integrated with techniques for conceptual schema integration and methods for requirements discovery and decomposition which provide a starting point for viewpoint detection. The mechanisms also detect situations which trigger process guidance for activities such as negotiation (Boehm et al. 1994) and validation (Maiden & Sutcliffe 1994). All these techniques and mechanisms will be integrated within NATURE's process model (Jarke et al. 1993) to enable active guidance for requirements engineers. It will also enable further integration of other research directions as a basis for a genuine, integrated solution for distributed requirements engineering.

## Acknowledgements

This research is funded as part of the European Union ESPRIT III 6353 'NATURE' basic research action.

## References

Boehm B., Bose P., Horowitz E. & Lee M-J., 1994, 'Software Requirements as Negotiated Win Conditions', Proceedings of IEEE Conference on Requirements Engineering, IEEE Computer Society Press, 74-83.

Jackson M. & Zave P., 1993, 'Domain Descriptions', Proceedings of IEEE Symposium on Requirements Engineering, IEEE Computer Society Press, 56-64.

Jarke M., Bubenko Y., Rolland C., Sutcliffe A.G. & Vassiliou Y., 1993, 'Theories Underlying Requirements Engineering: An Overview of NATURE at Genesis', Proceedings of IEEE Symposium on Requirements Engineering, IEEE Computer Society Press, 19-31.

Jarke M., Gallersdorfer R., Jeusfeld M.A., Staudt M. & Eherer S., 1994, 'ConceptBase - A Deductive Object Manager for Meta Data Bases', Journal of Intelligent Information Systems 3, 1-27.

Kotonya G. & Sommerville I., 1992, 'Viewpoints For Requirements Definition', Software Engineering Journal 7(6), 375-387.

Leite J.C.S.P, & Freeman P.A., 1991, 'Requirements Validation Through Viewpoint Resolution', IEEE Transactions on Software Engineering 17(12), 1253-1269.

Maiden N.A.M. & Sutcliffe A.G., 1994, 'Requirements Critiquing Using Domain Abstractions', Proceedings of IEEE Conference on Requirements Engineering, IEEE Computer Society Press, 184-193.

Maiden N.A.M. & Sutcliffe A.G., 1992, 'Exploiting Reusable Specifications Through Analogy', Communications of the ACM. 34(5), 55-64.

Nuseibeh B., Kramer J. & Finkelstein A., 1993, 'Expressing the Relationship between Multiple Views in Requirements Specification', Proceedings of 15th International Conference on Software Engineering, IEEE Computer Society Press, 187-196.

Spanoudakis G. & Constantopoulos P., 1994, 'Measuring Similarity Between Software Artifacts', Proceedings of 6th International Conference on Software Engineering and Knowledge Engineering (SEKE'94).

mappings between object attributes if both instantiate the same class of attribute. The minimum distance isomorphism is then used to select the best-fit isomorphism for all attribute mappings between two viewpoint models. Estimation of attribution distances is achieved through recursive generation of optimal isomorphic mappings between all object attributes. Therefore, this mechanism uses both structural and semantic similarities between components to detect analogical mappings between viewpoint models.

### 6.3: Example

This example demonstrates how similarity analysis detects isomorphisms which represent mappings between different components in two viewpoints for the lending library domain. Both viewpoints are expressed using a hypothetical object-oriented model, and both model the relationship between a borrower and a book in a library, but in different ways. The first viewpoint model (*Specification1* in Figure 5), expresses the employment relation by the object type *S1Borrowing*, while the second (*Specification2*) by the object attribute *Borrows*.

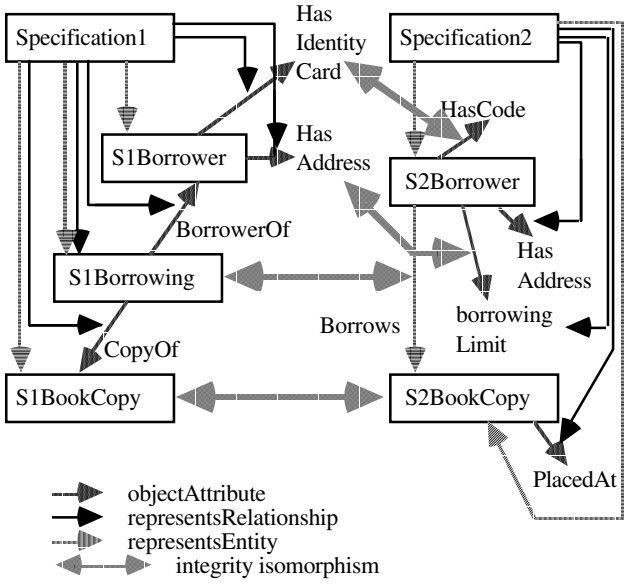


**Figure 5 - isomorphisms between specifications belonging to the same viewpoint model**

In addition to the structural information shown in Figure 5, both viewpoint models' components are classified using classes in the meta-model. Figure 5 also shows the isomorphism between components belonging to the two viewpoint models. It is important to note that components representing book copies, borrowers, their unique identifiers and the borrowing relations in both viewpoint models are all mapped. The mappings were derived using the minimum distance isomorphism. For example, *S1Borrower* is mapped to *S2Borrower* because both are the most similar pair of entity modeling components. Both had the same classification using the meta model (*Object-Type*, *Natural-Kind-Component*, *Entity-Modeling-Component* and *Concept-Modeling-Component*). Alternatives for *S1Borrower*, such as *S2BookCopy*, were classified as a nominal rather than natural kind component, so that the classification distance of the former pair was less than that for the latter pair. Furthermore, *S1Borrower* and *S2Borrower* have fewer structural differences than *S1Borrower* and *S2BookCopy*. The attribute *borrowingLimit* of *S2Borrower* has no analogous counterpart in *S1Borrower*. On the other hand, the attributes of the other entities had no counterparts.

### 6.4: Advantages & Further Research

The strengths of this approach are its similarity estimation mechanism and domain-independence. It can detect discrepancies between viewpoints using meta-knowledge which can be independent of the specification modelling notation and underlying domain, so it does not require domain analysis. This is an advance over existing computational mechanisms (e.g. Leite & Freeman 1991). Further research will include an elaboration and possible extension of the meta-model with further concepts which are domain- and specification notation-independent, formalisation of current and future versions of the meta-model, and large-scale empirical validation of this approach. Furthermore, the similarity mechanism scales up to handle viewpoints in real-world specifications. Its robustness has been shown through implementation and evaluation on a large specification object base (Spanoudakis & Constantopoulos 1994). Again, this is in contrast to existing techniques for viewpoint analysis which have neither been shown to scale to real-world problems nor how this might be achieved in the future.

## 7: Conclusions and Future Research

This paper introduces an integrated, albeit partial solution for distributed requirements engineering. It differs from existing research in its acceptance of complex definitions of viewpoints and emphasis on computational, cooperative assistance rather than method guidelines. It combines research from artificial intelligence, knowledge representation, domain modelling and deductive data base design to solve what is, in essence, a multi-disciplinary problem. Previous research has tended to address single facets of the problem such as viewpoint definition (e.g. Kotonya &

comparison of viewpoints with few if any syntactic differences. This is expected to scale to real-world requirements engineering problems, in contrast to existing lexical approaches. Although this solution can appear to be complex for the discrepancies detected in the example, its strength is for large viewpoints encountered in real-world practice.

## 6: Reasoning About Similarities Between Viewpoint Models

The third research direction uses reasoning about similarities to detect discrepancies between viewpoint models. Discrepancies such as contradictions and gaps occur between different requirements or between specifications of the same requirements which are modelled using different notations such as relational, entity-relationship or object-oriented data models, see Spanoudakis & Constantopoulos (1994).
Similarity analysis is achieved through classification of components in viewpoint models using a meta-model of modelling concepts. This meta-model is independent of application domains and expresses general semantic properties of models, thus it complements the domain modelling approach. Components in viewpoint models are classified using this meta-model during the requirements modelling process. Therefore, during similarity analysis, two components in two viewpoint models are assumed to correspond if these components share the same or similar semantic properties and have similar structures. This assumes that these components describe the same underlying real-world entities and express the same requirements.

### 6.1: The Meta-Model

The meta-model is expressed using the O-Telos notation, and has two parts. The first part contains taxonomies which group viewpoint model components according to modelling concepts which are independent of both the domain and specification notations (Spanoudakis 1994). In turn, two taxonomies enable component classification and hence viewpoint analysis. The first is organised as a generalisation graph of classes which group components according to general properties of real world concepts. The most general class *Concept-Modelling-Component* is specialised into *Entity-Modelling-Component* and *Relation-Modelling-Component*. Furthermore, *Entity-Modelling-Component* is specialised into *Natural-Kind-Component*, *Nominal-Kind-Component*, *Place-Component*, *Event-Component*, *State-Component*, *Activity-Component* and *Action-Component*.

*Binary-Relation*, a specialisation of *Relation-Modelling-Component* is specialised into 19 further subclasses which express groupings of binary relations according to cardinality constraints, general mathematical properties, existential constraints and semantic properties regarding related items.
The second taxonomy distinguishes between components which model relations in different viewpoints. Relation-Modeling-Components are classified using classes of different relation types such as *binary*, *symmetric*, *transitive*, *equivalence*, *set inclusion* or *set membership*.
The second part of the meta-model identifies generic constructs from different specification notations for underlying viewpoint models such as object-oriented or relational data models. For example, this meta-model includes a class grouping *object identifiers*, *types*, *attributes* and *isa* relations in a hypothetical object-oriented data model. This class specialises the more general classes of binary relations and relations with set inclusion semantics of the prescribed taxonomies. An example of these constructs for the relational data model is the class of inclusion dependencies, which also specialises the more general binary relations and relations with set inclusion semantic classes.

### 6.2: Similarity Analysis For Viewpoint Model Analysis

Each viewpoint model is an aggregation of components which are expressed as attributes of that model. Two different types of attribute are available for modelling viewpoints. The first attribute type expresses components which are entities and the second expresses components which are relations.
Analysis of similarities and differences between viewpoints uses a computed isomorphism between their components which reflects the analogical match between the viewpoint models. In this sense, it is similar to the pattern matching outlined in section 5. Computation of similarities is based on three metric functions which are the classification, generalisation and attribution distances between objects, detected from classification, generalisation and attribution relations in the O-Telos notation.
Classification distances determined between two viewpoint models indicate the semantic difference between these models according to our meta-model. Attribution distances generate an isomorphism between two viewpoint models which represent common structural similarities in the two models regardless of their representation. This isomorphism is generated using semantic homogeneities and the minimum distance isomorphism. Semantic homogeneities determine

component describing a viewpoint model is typed (e.g. object, state transition, event) according to our domain modelling language. Second, a powerful pattern matching mechanism only maps those components of the same type which also support the existence of an interrelated knowledge structure shared by the viewpoint model and each object system model (Maiden & Sutcliffe 1994). Therefore, components are mapped using their structural fit with other components in each viewpoint model.

Two results are possible for each pair of viewpoint models. First, different problem abstractions are retrieved for each viewpoint model, indicating little shared domain knowledge. Second, the same object system models are retrieved for different viewpoint models, indicating shared domain knowledge. In the second case another computational mechanism, called the viewpoint analyser, uses computed mappings to detect overlaps, gaps and contradictions. Detection is achieved using a small ruleset which exploits results from the domain matcher. The analyser has been implemented in Prolog. The basic functions of the mechanism are demonstrated in the following example.

## 5.2: The Example

Consider the retrieval of problem abstractions for our example library lending domain. Requirements engineers wish to integrate two models in viewpoints held by counter staff (VP2) and system administrators (VP3) through generation of a new viewpoint VP4. Therefore, the two models are duplicated and placed in the new viewpoint, similar to the procedure shown in Figure 1, to enable viewpoint analysis. Figure 4 shows a subset of inferred mappings which result from retrieving the same object system model for the two viewpoint models which describe basic lending and returning activities.

Rules have been defined to inform detection of different types of discrepancies between viewpoints. Examples of the main types of rule are given.

**Overlaps between viewpoint models:** components with the same names in two viewpoint models are mapped to the same object, behaviour, structure, goal or constraint in the problem abstraction. For example, both viewpoint models include a state transition labelled as 'returning' (same syntactic label), and both transitions have been mapped to the 'return behaviour' transition in the resource hiring system model.

**Gaps between viewpoint models:** no component in any viewpoint model is mapped to any object, behaviour, structure, goal or constraint in a retrieved object system model. Gaps indicate incompleteness which cannot be detected by existing computational mechanisms for viewpoint analysis. For example, in Figure 4, neither viewpoint model has a component which maps to the stative condition controlling the limit of loaned books to borrowers.

**Contradictions between viewpoint models:** syntactic and semantic contradictions are detected from inconsistent mappings between objects, behaviour, structure, goals and constraints in different viewpoint models and the same knowledge types in the object system model. One example of a syntactic difference in Figure 4 is that counter staff describe clients as 'borrowers' while the system administrators describe them as 'students', despite both being mapped to the client object in the resource hiring system model. Semantic contradictions are detected from complex mismappings with domain semantics in one object system model. Alternative solutions to this problem have been put forward, however our combination of mapping using domain semantics and pattern matching can provide greater leverage for detection of contradictions.

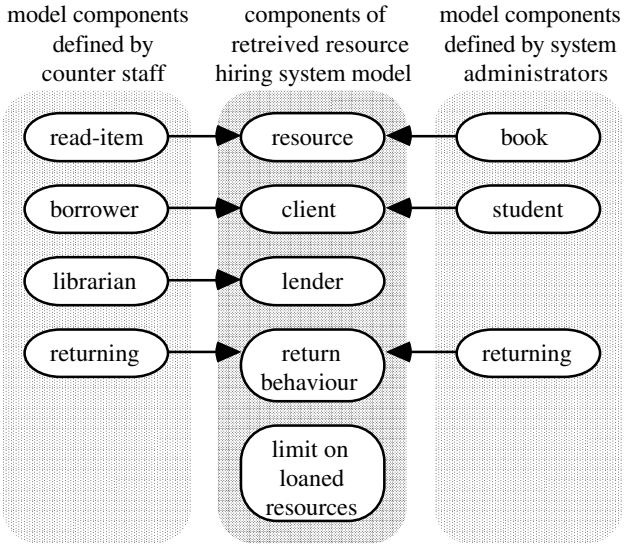


**Figure 4 - mappings used by the viewpoint matcher to detect discrepancies between viewpoint models**

## 5.3: Advantages and Future Research

Mapping different viewpoint models to the same object system models enables the use of domain knowledge to detect overlaps, gaps and contradictions without the domain modelling cost of other approaches (e.g. Leite & Freeman 1991). It also overcomes the need for domain-specific thesauri which are expensive to build and maintain. Its solution is to introduce complex computational reasoning mechanisms to the viewpoints problem for the first time. The use of pattern matching and mappings using domain semantics enables

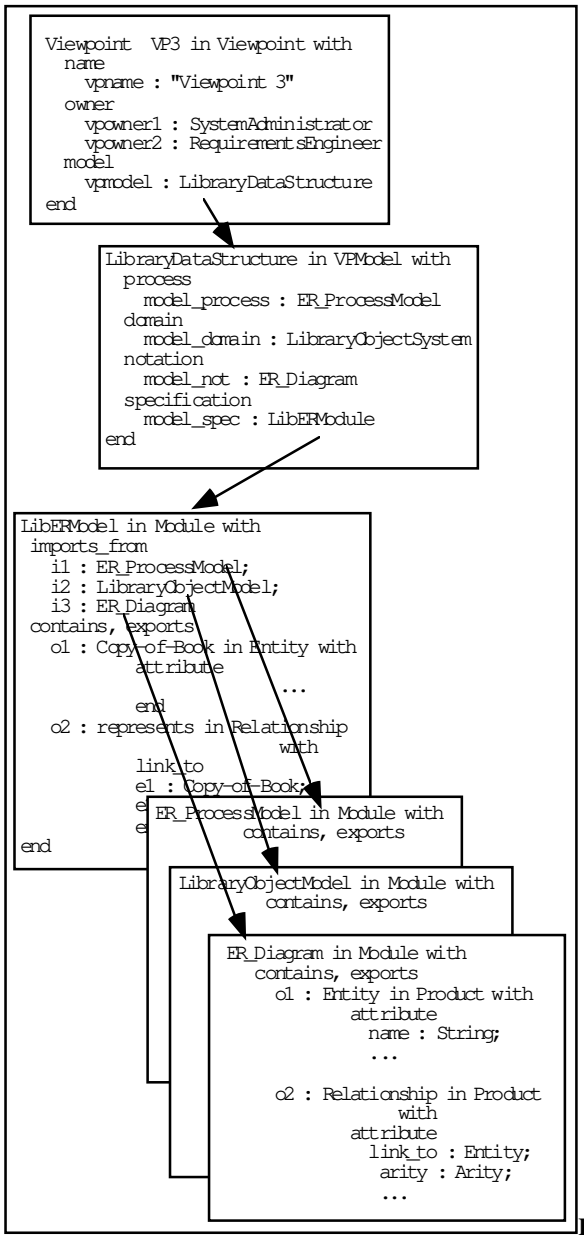


**Figure 3 - graphical representation of one viewpoint with 3 viewpoint models**

## 4.4: Advantages and Future Research

The distributed data base has been designed to handle the complexities of distributed requirements engineering by separating how viewpoints are used in the requirements engineering process from models which encapsulate domain, specification and representation knowledge. As such, it both extends current work and provides a platform for further research. The remainder of the paper describes two novel computational mechanisms for reasoning about the semantic contents of imported models to assist their analysis and integration.

## 5: Domain Knowledge for Viewpoint Model Analysis

Problem abstractions are models of the fundamental domain behaviour, structure, goals and constraints of all instances of one class of requirements engineering problem (Maiden & Sutcliffe 1994) held in a hierarchical data base. Viewpoint models can be matched to retrieved abstractions to enable their analysis. Indeed, abstractions focus reasoning on the fundamental features of a domain as the starting point for larger-scale viewpoint analysis and integration. This approach aims to resolve discrepancies between descriptions in different viewpoint models, therefore analysis is restricted to between viewpoints represented using our domain modelling language. The language is also founded on the O-Telos data model. The problem abstractions are also stored in the ConceptBase data base.

## 5.1: Problem Abstractions

Problem abstractions are composed of object system models. Each model is defined using state transitions triggered by events and controlled by agents, the structure of the domain, goal states, stative preconditions on state transitions, and object properties. A typical lending library domain is an instantiation of a number of object system models including resource hiring (instantiates to book circulation), resource repairing (book repair), resource provision (book purchasing) and object sensing (exit security system). Therefore, each viewpoint is owned by one or more agents and defines, in its models, partial domain behaviour, structure, goals, constraints and objects which instantiate one or more object system models.

Discrepancies are detected from mappings to object system models retrieved for each viewpoint model by the domain matcher (Maiden & Sutcliffe 1994). The matcher is a computational analogical reasoning mechanism which retrieves from 200 object and 20 information system models representing abstractions of business and real-time domains. It also computes mappings between components in each viewpoint and retrieved abstraction. These mappings then are used to compute overlaps, gaps and contradictions between viewpoint models.

The domain matcher combines pattern matching with mappings inferred from domain semantics to retrieve object system models. It can therefore detect mappings between components which have no syntactic similarities without using domain-specific thesauri or application models which are expensive to generate and maintain. The algorithm is in two parts. First, each

models, rather than core attributes such as owners, which are transferred between viewpoints.

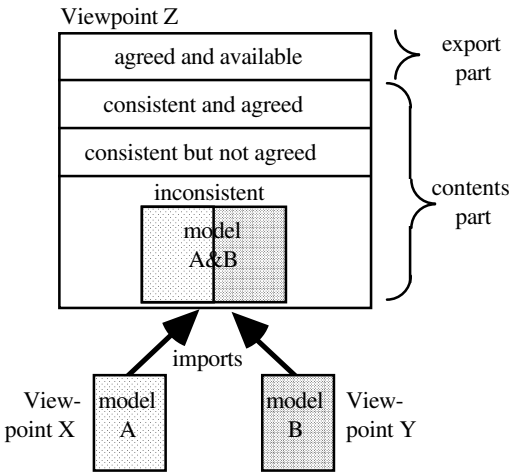


**Figure 2 - the process of viewpoint integration**

The definition presented in this paper also recognises that integration of viewpoints takes place throughout the requirements engineering process. One example of such integration is shown in Figure 2. A new viewpoint Z, consisting of a name, version, role, owners and process trace and guidance, imports several models from two existing viewpoints X and Y. Each model must change from inconsistent to consistent according to their syntax and semantics (the focus of mechanisms described in this paper). Once consistent, the semantic contents of the model must be agreed by all owners of the viewpoint, otherwise further negotiation and changes might be needed. A consistent and agreed model can then be made public, in which case it can be imported into other viewpoints for further integration.

## 4.2: A Definition of Modules

O-Telos has been extended using a notion of modules for representing viewpoints and viewpoint models. Modules are logically-distributed, virtual object bases which operate in a global data base and interact and communicate by import/export relationships. Each module must manage its own name space. This allows different objects to have the same name but in different modules, and permits imposition of different integrity constraints within the same viewpoint. This enables a framework for storing and reasoning about viewpoints and their relationships, as well as reuse, maintenance and integration of viewpoints.

Each module is divided into two parts to support the integration of models shown in Figure 2. The contents part represents knowledge which is private to a module and therefore only contains models available to owners of the viewpoint. The export part contains knowledge which can be imported by all other modules in the data base.

Each module is a standard O-Telos object which describes the contents of a module. In general, a module can be interpreted as a container of objects. All the objects defined for a module are connected to it via the `contains` attribute. Modules can also share objects, although access to objects defined in another module only if this module explicitly allows it. This results in the following definition of a module using the O-Telos notation:

```
Module with
  attribute
      contains : Object;
      imports_from : Module;
      exports : Object
end
```

A module accesses objects defined in other modules using the `imports_from` attribute and allows objects to be accessed by other modules using the `exports` attribute. Viewpoints and viewpoint models are also defined using the O-Telos notation:

```
Viewpoint with
  attribute
      name : String;
      version : String;
      owner : Participant;
      process : VPprocess;
      trace : VPtrace;
      model : VPModel
end
```

```
VPModel with
   attribute
      process: ProcessModel;
      domain:
ObjectSystemModel;
      notation: Notation;
      specification: Module
end
```

## 4.3: The Example

Consider the data base definition of viewpoint VP3, see Figure 3. This viewpoint has a name, version, role, and two owners. It also has three models which are implemented as separate data base modules. Figure 3 shows a Telos definition of the model containing the entity relationship diagram.

constraints on the syntactic and semantic consistency can be applied.

These definitions are demonstrated using an example which identifies different viewpoints for a new computer system for a lending library. The example is based in part on interviews with librarians from two real world libraries:

- a first viewpoint (name=VP1, version 1) has one owner (head librarian) and process guidance specific to its role in the requirements engineering process (a policy document). It has one viewpoint model containing requirements expressed in natural language;
- a second viewpoint (name=VP2, version 1) has four owners (counter-staff-1, counter-staff-2, counter-staff-3, sub-librarian-1) and process guidance specific to its role (a requirement model). At the start, several simple viewpoint models are defined to allow the four owners to work on each model, either in isolation or together as required. One contains a data flow diagram and is typed as such to enable application of constraints defined using the O-Telos notation. A second uses natural language while the third uses entity-relationship diagrams;
- a third viewpoint (name=VP3, version 1) has two owners (system administrator-1, requirements-engineer-1) and process guidance specific to its role (a requirement specification), see Figure 1. At the start, three simple viewpoint models are defined which include natural language descriptions and first-order logic rules. The third contains an extended entity-relationship diagram for the data base, and model type brings a set of constraints defined in the O-Telos notation.

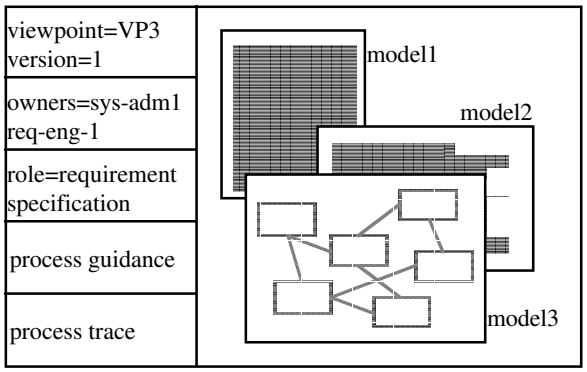


**Figure 1 - visualisation of viewpoint VP3, containing several viewpoint models**

These definitions present two challenges for our research. First, the data base definition must recognise the complex nature of viewpoints on requirements. Second, computational mechanisms must enable semantic reasoning between imported models in the same viewpoint to detect discrepancies prior to viewpoint integration. It is our intention that computational mechanisms aid viewpoint analysis, and provide information in the shape of detected discrepancies, to inform viewpoint integration, although it is important to recognise that integration also involves many human activities such as negotiation and agreement between stakeholders (e.g. Boehm et al. 1994).

## 4: The Viewpoint Data Base

Computational mechanisms require a common language and architecture to enable reasoning about viewpoints. This section outlines a language and architecture for storing, changing and reasoning about viewpoints and their models.

All viewpoints and models are represented and managed by the object base management system ConceptBase which uses the O-Telos data model. O-Telos was chosen as the underlying representation because it offers required representation concepts beyond standard data base applications, for example access to multiple versions of objects (O-Telos offers historical queries), multiple instantiation and specialisation of representation concepts and multiple instantiation levels of these concepts. The O-Telos model also supports assertions to declare integrity constraints and deductive rules in many-sorted, first-order logic. The remainder of this section concentrates on extensions to the data model which support the definition of viewpoints.

### 4.1: Viewpoints, Models and Modules

Each viewpoint is defined and stored in the data base in two parts. First, the core viewpoint contains information about the viewpoint name, version, owners, roles, process guidance and process trace which are used to manage the viewpoint in the requirements engineering process. The core is implemented as a standard Telos object, with attributes for name, version, owners and role and relationships with process trace and process guidance. Second, each viewpoint is connected to viewpoint models, each of which maps to a data base module. A viewpoint model is typed according to its representation or language to enable retrieval of predefined objects applicable to that model. Therefore, the core enables the viewpoint to be manipulated as an artefact in the requirements engineering process, and points to models belonging to that viewpoint. Viewpoint models enable the data base to handle complex viewpoints which are produced at different rates and with different levels of formality. It is these viewpoint

process guidance and requirements tracing (Jarke et al. 1993).

The paper has four parts. First, the three research directions are outlined. Second, viewpoints on requirements are defined. Third, each research direction is detailed and advantages for requirements engineering are described. Future research is then discussed.

## 2: Computational Mechanisms for Distributed Requirements Engineering

Our first research direction investigates how to store viewpoints in requirements engineering repositories. There has been little recognition of problems of storing, updating and integrating viewpoints on requirements. The paper defines a data base for this purpose. Advantages from it are at least two-fold. First, it emphasises the problems which arise when handling viewpoints, and second, it provides a basis for design of computational mechanisms for reasoning about viewpoints. All viewpoints are represented and managed by the object-oriented data base ConceptBase which uses the O-Telos data model (Jarke et al. 1994). O-Telos is extended using a specialised notion of modules to represent viewpoints. Modules are logically-distributed, virtual object bases which operate in a global data base and interact and communicate by importing and exporting objects. This enables a framework for storing and reasoning about viewpoints and their relationships, as well as reuse, maintenance and integration of viewpoints.

A second direction retrieves problem abstractions (Maiden & Sutcliffe 1992) to assist viewpoint analysis. Problem abstractions represent the fundamental behaviour, structure, goals and constraints of all instances of one class of software engineering problem. These kernels of domain knowledge enable detection of overlaps, gaps and inconsistencies between viewpoints by matching different viewpoints to the same problem abstractions.

The third research direction uses meta-modelling and similarity analysis to detect discrepancies between viewpoints (Spanoudakis & Constantopoulos 1994). It exploits computational mechanisms for analogical reuse of requirement specifications. A meta-model of knowledge types which is independent of domains and specification modelling notations detects isomorphisms as a basis for viewpoint integration where appropriate.

These research directions are integrated within the ESPRIT NATURE basic research action (Jarke et al. 1993). Domain modelling and reasoning about similarities complement each other. Problem abstractions are common artefacts which enable computational reasoning between viewpoints about the fundamental features of the problem. Reasoning about similarities provides reasoning mechanisms to detect discrepancies between viewpoints. Its strength is its domain-independence, so it is applicable to large requirement specifications with many viewpoints without the need for domain modelling. Both research directions are built on a common knowledge representation language and data base. This provides the framework for design of the computational mechanisms, while design of the mechanisms impose language representations on the representation used by the data base. The data base also supports creation, analysis and integration of complex viewpoints as well as imposing constraints and guidelines for the analysis process. The result is a complex and integrated, but still partial solution to support distributed requirements engineering activities.

This paper provides an overview of this solution, although space limitations mean that details of each research direction are found elsewhere. First however, a starting definition of viewpoints on requirements is provided as a basis for the work presented in this paper.

## 3: A Definition of Viewpoints in Requirements Engineering

NATURE adopts a broad definition of a viewpoint in requirements engineering. In essence it claims that a viewpoint is any artefact produced as part of the requirements engineering process, through which its owners communicate and agree. However, this definition is insufficient to inform design of storage and reasoning mechanisms. Therefore, a second definition which focuses on the knowledge held in each viewpoint was produced. Since our aim is to provide computerised support for distributed requirements engineering, this second definition focuses on contents which might be held in electronic format in a data base. A viewpoint is defined as a loosely-coupled, locally managed object encapsulating different types of knowledge about requirements. The principle differences with Nuseibeh et al's (1993) definition are that each viewpoint can have different representation schemes and multiple owners with different relationships to the viewpoint. Each viewpoint has: a name and version number, owners (stakeholders); a role which guides the reader to interpret the semantic contents of a viewpoint; the trace of decisions, actions and contexts in which decisions are taken from generation to completion of the viewpoint; and process guidance for extending, validating and integrating the viewpoint; one or more viewpoint models which express knowledge about the domain or system using one or more representation schemes to which

# Computational Mechanisms for Distributed Requirements Engineering

N.A.M. Maiden[1], P. Assenova[2], P. Constantopoulos[4], M. Jarke[3], P. Johanneson[2], H.W. Nissen[3], G. Spanoudakis[4] & A.G. Sutcliffe[1]

[1]Centre for Human-Computer Interface Design, City University, London
[2]Swedish Institute for Software Development, Kista, Stockholm
[3]Lehrstuhl fuer Informatik V, RWTH-Aachen, Aachen
[4]titute of Computer Science, Foundation of Research and Technology, Crete

## Abstract

*This paper argues for new computational mechanisms to aid specification of requirements for composite systems. It presents mechanisms for storing specification fragments, or viewpoints, and reasoning about these viewpoints to resolve semantic discrepancies prior to their integration in a comprehensive solution for distributed requirements engineering.*

## 1: Introduction

Requirements for complex, composite systems are acquired and modelled from different and distributed viewpoints (Nuseibeh et al. 1993) held by participants who produce the specification. This is known as distributed requirements engineering. In most cases these viewpoints must be integrated to provide either a contract for design during requirements procurement or a specification for bespoke system design. Overlaps and gaps must be identified and expressed, participants made to interact and cooperate, and contradictions resolved. However, there has been little computational assistance for analysing and integrating these viewpoints, despite its necessity when handling large-scale requirement specifications. This paper describes two novel computational mechanisms and a distributed data base design which have been designed as part of the ESPRIT NATURE basic research action (Jarke et al. 1993) to provide part of a comprehensive solution for distributed requirement engineering.

There has been little research into computational mechanisms for reasoning about viewpoints. These mechanisms must, among other things, reason using different representations and detect semantic differences as well as terminological differences for the same constructs between viewpoints. Paper-based methods such as VOA (Kotonya & Sommerville 1992) provide guidelines for detecting and notations for modelling viewpoints but do not aid analysis and integration. This paper argues that viewpoints on requirements are large and complex and need computational mechanisms to aid analysis and integration.

Reasoning mechanisms which do exist are simple and use domain and specification syntax rather than semantics. Thus, differences between names in viewpoints can be detected, however more complex reasoning is impossible. For example, viewpoint-oriented software development exploits a set of inter-viewpoint rules which detect inconsistencies using syntactic names of specification objects including notation features and diagrams (Nuseibeh et al. 1993). Leite & Freeman (1991) proposed a similar set of heuristics to perform syntactic viewpoint analysis.

Furthermore, requirement specification languages tend to model single, unified perspectives, although some exceptions exist (Jackson & Zave 1993). In distributed requirements engineering, these languages must impose local consistencies within viewpoints but permit global inconsistencies between viewpoints. In addition, people work apart but access a common data base containing the requirement specification and definitions of commonly-used models such as problem abstractions (Maiden & Sutcliffe 1994), process models (Jarke et al. 1993) and notations, thus posing additional problems for architectures which support these languages.

This paper proposes three novel and interrelated research directions within the ESPRIT NATURE project (Jarke et al. 1993) as solutions for distributed requirements engineering, namely domain modelling (Maiden & Sutcliffe 1994), similarity analysis (Spanoudakis & Constantopoulos 1994) and deductive data base design (Jarke et al. 1994). These directions are incorporated in a wider framework to guide the requirements engineering process which encompasses