CREWS Report 97-11

View-Directed Requirements Engineering: A Framework and Metamodel

R. Motschnig-Pitrig, H.W. Nissen, M. Jarke

appeared in
Proc. of the 9th Intl. Conf. On Software Engineering and Knowledge Engineering (SEKE '97),
Madrid, Spain, June 17-20, 1997.

View-Directed Requirements Engineering:
A Framework and Metamodel

Renate Motschnig-Pitrik
Applied Computer Science and Information Systems
University of Vienna
Rathausstr. 19/4, 1010 Vienna, Austria
renatem@ifs.univie.ac.at

Hans W. Nissen, Matthias Jarke
Informatik V
RWTH Aachen
Ahornstr. 55, 52056 Aachen, Germany
{nissen,jarke}@informatik.rwth-aachen.de

Abstract

Semi-formal methods for requirements and design suffer from a problem of scalability. They offer different notations to describe complementary abstractions of the problem to be analyzed, but each such abstraction in itself becomes rapidly much too big to be handled in one piece. Some methods therefore offer basic view mechanisms, variously called modules, categories, viewpoints etc., to introduce smaller units. However, they lack in guidance what views to define and how to interrelate them in a systematic manner. The problem we address in this paper can be simplistically stated as: "Given that a multi-notation specification must be composed from hundreds of fragments each no larger than one page, according to what facets should we acquire, classify, compose/ decompose, and compare these views?" We develop a situational framework of such facets, and formalize it in a metamodel. We compare three basic process models of how the framework can be applied, and describe the formal and technical support required for each of these models.

1 Introduction

The early phases are gaining recognition as cornerstones of the software process. However, the complex nature and large scale of today's systems requires proper means to organize large amounts of diversified and heterogeneous information. A well known means of achieving a transparent organization is that of *abstraction* [Shaw84]. The classical abstraction mechanisms of classification, generalization, and decomposition organize concepts based on *intrinsic properties* such as similarity, spatial and temporal vicinity, and inclusion [Motschnig-Pitrik96].

A complementary strategy of building abstractions, based on *extrinsic grounds*, considers the viewing of some subject matter in different *situations* or from different *perspectives*. Although the resulting abstraction, referred to as *viewpoint abstraction*, has rarely been considered along with classical abstraction mechanisms. A recent survey shows that its usefulness in building models of the real world is well established in software development, information systems, and AI for more than two decades [Motschnig-Pitrik95]. However, only within the 1990's viewpoint abstractions are systematically investigated, formalized, and supported by automated tools.

In this paper, we are interested in using the viewpoint abstraction in order to solve a particular problem in requirements engineering (RE), namely that of **scalability** and **viewpoint organization**. Imagine you have to develop a large requirements specification according to some well-established methodology such as ER/SA or OMT. The PC on which your CASE tool runs has only a limited screen real estate available which can show just a few small views on the specification at the same time. The following questions then arise:

1) What should be the heading of each window so that you understand its role in the overall specification? This is equivalent to ask: according to what *criteria* do you partition the overall specification into views?
2) To what degree, and in what way, should the different views *overlap*, such that awareness of mutually relevant changes is ensured in the development team? How can controlled redundancy elicit productive conflicts in requirements elicitation?

In section 2, we address the first question by proposing a *framework of situation parameters* according to which views can be classified. We then map this framework into a metamodel which can serve as a basis for the structuring of software information bases.

In section 3, we validate the framework by showing how a diverse set of methodologies (a traditional one, a viewpoint-oriented one, and an object-oriented one) can be expanded by it, respectively which parts of the framework are already captured in them.

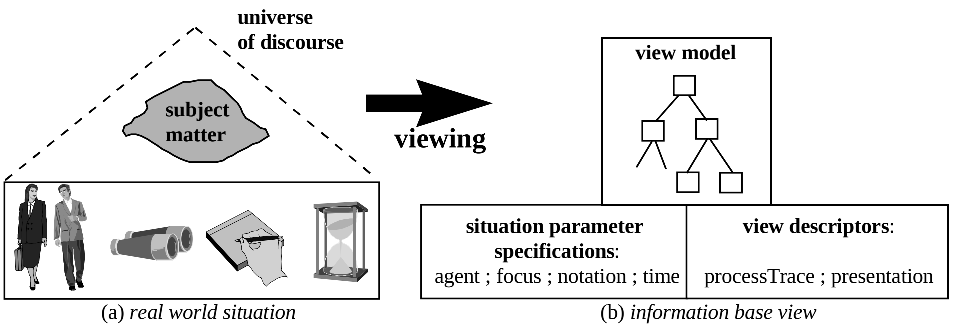

Figure 1: Mapping real world situations into information base views

In section 4, we investigate *systematic ways to address the second question*, i.e. possible strategic frameworks behind viewpoint resolution techniques discussed in the literature. Since we propose that the acquisition of views according to a predefined framework shall direct the process of RE, we refer to our approach as *view-directed requirements engineering (VDRE)*.

Finally, in section 5, we show how our approach complements related work in cooperative information systems engineering and discuss directions of further research. Throughout, the example of a 'university library system' is used to illustrate individual issues.

2 Mapping Situations To Views

We assume that VDRE proceeds within a particular universe of discourse (UofD). The UofD comprises the system to be developed, its environment, and everyone and everything related to the system in some information-revealing way, such as users, management, the software development team, customers, etc.

For example, the UofD of a university library system encompasses the university as the covering organization, the people responsible for the working of the library, all activities ensuring the proper working of the library, potential customers, available hardware, etc. It is understood that the borderline between the UofD and its environment is not a rigid one. It is subject to alteration throughout the view-directed requirements engineering process. This is indicated in figure 1 by drawing a dashed line to delimit the scope of the UofD.

A *view* is a complex data structure holding a specification fragment that is intended to be stored in an Information Base (IB). A view's purpose is to capture information on some well defined slice of the UofD (which we call *subject matter*) under well-defined conditions (which we will call *situation parameters*). Aside from having a name, a view consists of three major constituents: the *view model*, the *specification of the situation parameters*, and the *specification of the view descriptors*. The latter are the *process trace* and a set of potential *presentations* of the view model. The process trace parameter captures the view's version, state, and history and, in particular, the sequence of operations that led to the current version. The semantics of a view is such that the *view model* is a named specification of the subject matter that is expressed in a specific notation and characterized by the situation parameters.

2.1 Situation Parameters

A viewing situation refers to some *subject matter* and is characterized by a number of factors that describe the particular circumstances of viewing (compare figure 1). However, before going into details, let us clarify the relationship between the information model notion of a *view* and the real world notion of a *situation*. The real-world situation can be conceived as being composed of some subject matter - the "what" of a situation - and a number of *situation parameters* - the "who", "why", "how", and "when" of the situation. They support the categorization of a information base view and play a major role in the strategy for view elicitation, integration and change management.

Subject Matter. Usually, a viewing situation consists of some *subject matter* that is part of the UofD. In general, a view's subject matter can be abstract or concrete in nature and may cover a major topic of the UofD or just a single object/concept thereof. As examples of the former category consider the conceived software system or any one of its components. Single objects/concepts of the UofD of the university library are exemplified by the object of a book and the concept of a librarian.

Agent. The *agent* parameter documents two kinds of roles: firstly, the person who actually views the subject matter - the *view holder* - and secondly the person who specifies the view holder's facts and beliefs- the *view engineer*. In certain cases the view holder may also be occupied by some external, automated entity that interacts with the target system [Kotonya92].

Focus. A view often has a particular purpose or topic, is biased towards some central aspect, and is oriented toward capturing a predefined level of detail. The *focus* parameter captures all these criteria by its attributes : The *topic* attribute captures the main theme of the view, e.g., systems analysis or quality assurance. The *aspect* attribute determines the major bias of a view, e.g., dynamic behavior or static structure. Finally, the *detail* attribute describes the level of detail of the view.

Notation and TimeFrame. The *notation* used for view specification is important because it acts like a filter in expressing ideas, by restricting the available modelling constructs -- "everything that cannot be expressed, must be left unsaid" (Wittgenstein). The *time frame* helps to delimit the 'validity' of a view to a particular point in time. This is necessary, in order to respect the evolutionary nature of the subject matter and the fact that agents are likely to change their opinion due to further insight acquired as time passes.

2.2 A Formal Metamodel

The parameters describing a particular viewing situation are summarized in the formal metamodel presented in figure 2, using the OMT notation. The concepts of OMT itself have been meta-modeled, e.g., in [Hong95]. This model offers a more detailed picture of the situation parameters discussed in the last subsection, based on the application of the framework to existing methods as reported in section 3. The extended model serves as a starting point for a general strategy on systematic view derivation and for a categorization and characterization of view model relationships. In particular, note the part-of relationships of *Focus*, indicating that, in general, the focus of a view model is determined by multiple parameters.

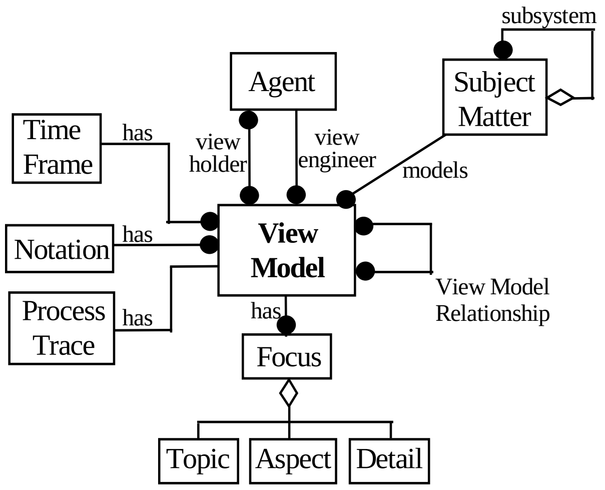

Figure 2: Metamodel of the view object model in OMT

Further note that the subject matter may be decomposed into subsystems, on which the view-directed approach may be applied recursively. Further structuring, not shown in the meta model, can be achieved by arranging agents and/or topics along generalization/ specialization hierarchies. The hierarchical structure may be exploited such that, for example, more specific topics inherit attributes from more generic ones or responsibilities may be implied for agent roles.

It is the constellation of several views that collectively form the requirements specification. Thus, the relationships between views are of extreme importance. Figure 3 depicts various kinds of relationships between view models derived from the situation parameters. In practice, two view models may differ with respect to one or more of their parameters.

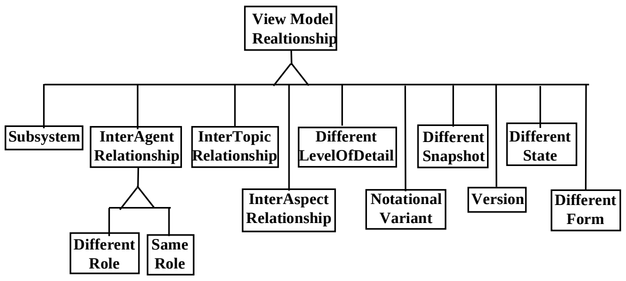

Figure 3: Specializations of view model relationships

3 Applications of the Framework

View-directed RE is not a radical departure from traditional software development techniques. Rather, it is intended to extend and complement them by taking into account a richer and well structured environment of factors influencing the conceived target system. In order to illustrate the evolutionary nature of view-directed RE, we briefly show how some well known methods can be mapped into our framework.

3.1 A Structured Method Example

One of the earliest and most broadly applied requirements analysis (RA) methods is SADT [Ross77]. SADT captures views along three dimensions:

- actigrams versus datagrams;
- levels of detail;
- alternative decompositions

A given topic can be analyzed according to the functional aspect using the actigram notation and according to the data structure aspect using the datagram notation.

This is illustrated in figure 4 which deals with the topic 'system requirements' and the 'processing of book requests' as subject matter. The left hand side of the figure depicts the aspect of 'activity decomposition', expressed as actigrams, whereas the right hand side uses the notation of datagrams to express the aspect of ‘data usage’.

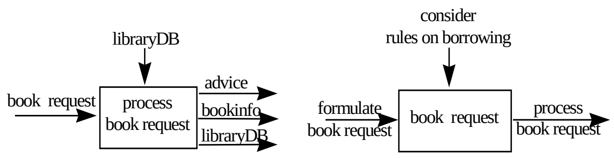

Figure 4: Aspects and notations in SADT

The decomposition of processes leads to the variation of the level of detail. In order to be consistent, the model on the more detailed level must have an interface that is a refinement of the parent diagram's interface. Figure 5 holds the decomposition of the activity ‘process book request’ and is therefore a variation of the level of detail.

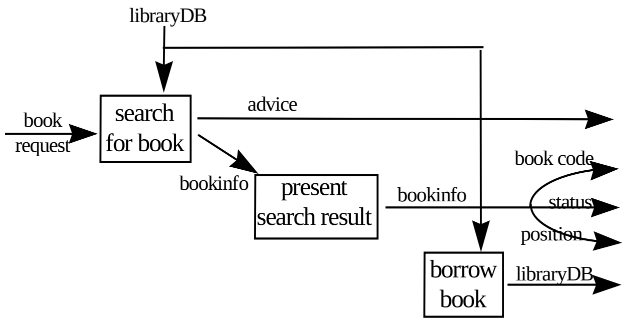

Figure 5: Decomposition of ‘process book request’

Aside from variations regarding the aspect and the level of detail, SADT allows to express different topics manifesting themselves as alternative decompositions. Thus, the activity ‘process book request’ according to the topic ‘usage’ (rather than ‘system requirements’ above) would lead to an alternative decomposition, as shown in figure 6. Note that SADT does not impose any integrity constraints on models with different topics.

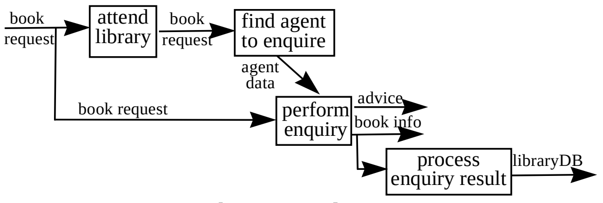

Figure 6: Alternative decomposition

3.2 A Viewpoint Method Example

CORE [Mullery79] is a functional requirements definition method based on viewpoints that are defined on two levels. The first level consists of all entities that interact or affect the system in some way. At this level, functional and non-functional viewpoints are distinguished. In the context of the library example, functional viewpoints are exemplified by student, librarian and the library database, whereas non-functional viewpoints are, for example, efficiency, extendability, reliability. On the second level, *defining viewpoints* are distinguished from *bounding viewpoints*. The former are subprocesses viewed in a top-down manner, the latter refer to entities that interact with the system. In terms of our framework, the functional and non-functional viewpoints could be seen as different *aspects* of the system requirements *topic* and each bounding viewpoint would be modelled as a *view holder*. Subprocesses correspond to different *levels of detail* of the functional requirements aspect, in analogy to the structured method example above.

3.3 An Object-Oriented Method Example

As an example from the field of OO methods, OMT [Rumbaugh91] proposes to capture views along four dimensions:

- the analysis and the design model are variations on the *topic*;
- the static model, dynamic model, functional model, and data dictionary are different *aspects*;
- all models, except for the data dictionary, can be viewed at different *levels of detail*;
- the 'data model' aspect is composed of models employing different *notations*, as, e.g., scenarios, event traces, event flow diagrams, and state charts.

An example for a relationship between a view expressed in different notations are the class and the instance diagram notation of the static model aspect. Of course, each instance diagram must conform to the specification contained in the class diagram. Sometimes, scenarios and statecharts or class and instance diagrams can be considered as different aspects, rather than just notational variants.

4 Processes and Environments for VDRE

Based on the discussion in the preceding sections, we first discuss the systematic construction of views based on the variation of situation parameters. We then compare three basic process alternatives for view-directed requirements engineering and investigate the necessary properties of supporting environments.

4.1 View Construction and Integration

In soliciting views, the primary objectives are to obtain views that:

- in their entirety provide full coverage of all relevant aspects of the subject matter;
- can easily be compared and integrated to form the requirements specification.

Often, the RE team will aim to arrive at a set of consistent requirements. However, during most of the process the view-directed approach allows them to proceed with alternative views holding inconsistent view models, as long as such inconsistencies are made apparent and it is agreed that they may be resolved at some later stage [Jarke96, Easterbrook96].

In the following, we discuss the effects and interdependencies resulting from the variation of individual view constituents. Subsequently, we derive a general strategy that leads to views that meet the objectives mentioned above.

Subject matter. First, it has to be decided whether views shall be elicited on the whole system or rather on subsystems. In the latter case the system must be decomposed such that the subsystems collectively cover the whole subject matter. Subsystems need not necessarily be disjoint since overlaps can be handled by the view mechanism and may indeed be productive for reasons of group awareness and conflict identification [Higa95, Nissen96].

Agents. The selection of view holders must take into account the coverage of the subject matter as well as the coverage of all topics and aspects considered important for the RE process. For this purpose, the distinction of agent roles may prove useful [Gotel94]. If more than one view holder is selected from one role, it is meaningful to associate the same view engineer with these agents such that enquiry-induced and notational differences are kept at a minimum; sometimes, one may also want to vary the view engineer in order to blend out enquiry-induced biases.

In general, varying view holders and keeping all other parameters constant, results in views that reflect the inter-agent (often inter-personal) differences of stakeholders, as a basis for analysis and reconciliation.

Focus. Aspects are related to notations in that there exists a number of special notations that are best suited for expressing particular aspects. For example, the notation of data-flow diagrams or bubble charts are suited to express the functional model of a system. If more than one notation is used to express the same topic and aspect (all other situation parameters being kept constant), the resulting models form notational variants, often motivated by the necessity to provide different addressees of views.

Notations. In some cases, transitions between different levels of detail imply transitions in notations. Data-flow diagrams, for example, allow for several level of detail transitions without notational changes, up to the point where simple processes are reached. The specification of these via pseudocode proves more effective than continuing to draw data-flow diagrams. Since notations act as filters on what can be expressed, they are highly dependent on other components of views and on the views' addressees.

Levels of Detail. This component serves to point to the expected relative information content of views, in order to facilitate their comparison. Bottom-up strategies start with views at a low level of detail and proceed only after reasonable consensus has been reached on that level; top-down strategies proceed by decomposition. In both cases, the comparison of inter-agent views with equal level of detail is of primary importance.

Time Frames. Due to the evolutionary nature of the real world and hence of both the subject matter and a view holder's perspective thereof, it has to be documented which snapshot a view model is to reflect and how this relates to earlier snapshots (if captured). Considering snapshots is a necessary prerequisite to change management. In the view-directed approach, changes to a view model that already has been checked for consistency with other views or that already has gone through integration processes, calls for a reiteration of the checking and integration process.

4.2 Process Approaches to View-Directed RE

Theoretically, a large number of possible ways-of-working within view-directed RE exist. However, looking at practical approaches three main lines of development processes can be distinguished: the *weakly guided approach* where only the rough direction is specified and anything else is left open, the *strongly guided approach* where the situation parameters and the view contents are completely fixed, and the *goal-oriented teamwork approach*, a compromise of the two extremes before, where the process steps and goals are collaboratively defined at the beginning of the project.

The **weakly guided approach** offers a strategy on view acquisition independent from the conrete application domain and development goals. The object-oriented [Booch91,Rumbaugh91] and structured analysis methods [Ross77] fall into this category. They offer a set of different languages to represent different aspects of the problem domain. The way-of-working is mostly limited to generic guidelines for applying the modeling constructs of the supported notations. In general, the objective is to cover all aspects of the problem domain by varying the situation parameters in a way that the individual views stay as orthogonal as possible. The application of this strategy enables a flexible and self-definable view development process.

A generic guideline to keep the developed views comparable in such settings is to vary as few situation parameters as possible at a time. Following the discussion in the preceding sections this can be stated as follows:

- Select different view holders of the same role keeping everything else constant.
- Select view holders of different roles and enquire them about topics/aspects they are knowledgeable about.
- Get views with respect to all topics/aspects. Vary agents only if necessary due to competence.
- Vary the level of detail, everything else being kept constant.

The **strongly guided approach** offers a set of predefined views as *reference models* for different problem domains (as, e.g., mechanical engineering, furniture factories). They describe the typical data, processes and functions within these sectors, together with a set of consistency constraints which define the syntactic and semantic relationships between the models. They are justified by experiences from successful analysis projects in these industrial sectors. In their original state the views satisfy all constraints. A provider of such reference models is SAP who ships *business blueprints* together with their business information and management system R/3.

The individual development project consists of the selection of the applicable view models in combination with a slightly variation. To ensure consistency, only the modified parts of the views have to be checked. The effort to analyze and model the problem domain is reduced to the comparison of the predefined view models with the actual reality and their customization [Scheer94]. However, this reuse of notations and views is less flexible than the weakly guided approach .

A compromise between the two extremes is given by the **goal-oriented teamwork approach**. At the beginning of a development project the specific analysis goals and, induced by them, the notations, topics and aspects of the views to be acquired are defined in group sessions. The view models will then be created in a parallel and independent, but goal-oriented way. The analysis goals define how the different view models relate to each other and therefore describe how the parallel development should be coordinated and how the resulting views can be integrated. Examples of such methods are IBM's JAD (Joint Application Design [August91]) and DEC's PFR (Analysis of Presence and Future Requirements [Abel95]).

The choice between these three approaches is determined by the trade-off between flexibility and efficiency. Weak guidance is very flexible since it does not constrain the development process; but it is only efficient in small projects because it gives no guidelines for the view acquisition process. Storng guidance enables an efficient analysis process but these guidelines in form of standard solutions cannot be easily tailored to non-standard cases. In the third approach the guidelines are developed in a common group meeting at the beginning of a project. This allows for flexibility in defining the analysis goals and efficiency in the acquisition of the views based on the goal-orientation. Combinations of the second and third approach are conceivable in large projects with standard and non-standard components.

4.3 Support Environments for VDRE

The three approaches to view-directed RE described above require different support technologies, for the management of individual views and for the resolution of multiple views.

Common to all is the need for a *separate representation* of the individual views to allow conflicts and inconsistencies between them. Differences occur in the degree of *integration* that is required by the methods and the management of global knowledge.

Weak guidance does not require any global management information. Multiple views can be independently developed in a completely distributed support environment. Such an environment is realized by the ViewPoints approach [Nuseibeh94]. Views are distributed over locally managed objects called ViewPoints which act as independent data bases without a central control mechanism. In [Mylopoulos95, Constantopoulos95] an alternative is presented where the views are managed within a central repository but separated by so-called contexts.

Strong guidance requires mechanisms for the selection and customization of the predefined view models, such as the ARIS-Toolset [ARIS96] for the modelling of business processes. Since the integration of these views as well as the consistency constraints are known in advance and the views are already prepared according to them, a central control instance is not needed.

For the goal-oriented teamwork approach, a special designated part for the representation of the analysis goals must exist. Since goals can vary across projects as well as within one project, their specification must be easily modifiable.

Very different are also the needs for *resolution techniques of the three approaches*. The first approach concentrates on the separation of view development. It employs no global resolution strategy or constraints. For every view the relationships to other views can be stated locally, such that a communication and evaluation interface available at all views is sufficient. Only a common language for consistency constraints is still necessary.

In the second approach, all applicable resolution rules are predefined and cannot be modified. This allows for an efficient realization in which the constraints are an integral part of the development environment. In ARIS, for example, these constraints are hard-coded within the system and work even in the case of a selective use of the reference models.

For the third approach, the view resolution process must be performed according to the defined analysis goals. Their modifiability requires extensible meta modeling facilities which are currently only offered by very few support environments [Mylopoulos90, Jarke95]. A development environment supporting the adaptable definition of resolution constraints and their monitoring is reported in [Nissen96, Nissen97]. The analysis goals are represented as a user-defined meta meta model common to all developed views which can be expressed according to different notations or domain meta models. The application of a model-based viewpoint resolution technique guarantees the analysis of the views according to the specified meta meta model.

5 Discussion

The basic hypothesis underlying the view-directed approach to RE is that the quality of large requirements specifications can be enhanced if (i) requirements are elicited from multiple sources with carefully controlled syntactic, semantic, and pragmatic redundancy and (ii) requirements are captured systematically such that information on particular topics is localized and structured.

In this paper we identified and formalized a number of factors, called *situation parameters*, that characterize the real world situations from which views are derived. We employed them for the systematic acquisition such that the resulting views can be easily compared and checked for consistency. We indicated in which way the proposed framework complements a range of existing analysis techniques. Furthermore we applied the framework in various process models and discussed the support requirements for each of the application scenarios.

Our view-directed approach appears complementary when compared with other viewpoint approaches.

Leite and Freeman (1991) define a viewpoint as a mental position used by an individual when examining a universe of discourse. Similar to us, they advocate a viewpoint resolution approach but do not consider explicitly the case of a very high number of viewpoints. The work by [Maiden95] does elaborate viewpoints along the three dimensions proposed by [Pohl94], together with some proposals for computational mechanisms supporting resolution. But the implicit assumption of that work is a small number of viewpoints to be reconciled, and no design guidelines for what views to choose are given.

ViewPoints [Nuseibeh94] start from the premise that complex systems are comprised of heterogeneous components whose requirements are specified using multiple methods and notations. ViewPoints are designed to hold partial requirements specifications, developed according to different strategies and specified in different representation schemes. The view-directed approach provides in addition guidance on the selection of views to be captured and on suitable strategies for view integration,

Whereas ViewPoints are more general than our approach, VOA [Kotonya92] appears more specialized. They incorporate a classification into active and passive viewpoints. In VOA, a viewpoint is defined as an external entity that interacts with the system being analyzed, but one that can exist without the presence of the system. Note that we adopt precisely this definition for what we call the ’agent’ component of a view. Consequently, all the insightful strategies for viewpoint elicitation, structuring and integration proposed for VOA are directly applicable within the view-directed approach.

Due to the genericity and the wide range of applications of the view-directed approach there exists a large agenda for further research and detailed case studies. We are in the process of applying and specializing the framework for use with OMT [Rumbaugh91] and with Objectory [Jacobson92]. A further challenging task is the specification of generic inter-view consistency rules based on view relationship categories. As an example for an inter-aspect relationship consider the task to establish under what conditions a user interface shall be consistent or compatible with a user model. We are also interested in strategies regarding the integration process, i.e., the order of steps to be taken to achieve the required level of consistency with a reasonable effort. In [Nissen97], a thorough methodology for at least the third process approach mentioned, that of goal-oriented teamwork processes, is elaborated and encouraging application experiences are reported.

Acknowledgements. This work was conducted while the first author was visiting RWTH Aachen. It was supported in part by the European Commission in Esprit projects 21903 (CREWS) and 22469 (DWQ), and by the European-Canadian cooperation grant ISC-CAN-080 CIS (EU/CAN CIS).

References

[Abel95] Abel P.: Introduction to the PFR Analysis Method. Technical Report, USU AG, Germany, 1995.

[ARIS96] ARIS-Toolset V3.1 Manual. IDS Prof. Scheer GmbH, Saarbrücken, Germany.

[August91] August J.H.: Joint Application Design: The Group Session Approach to System Design. Yourdan Press, Englewood-Cliffs, 1991.

[Booch91] Booch G.: Object Oriented Design with Applications; Benjamin/Cummings Publ. Comp., 1991.

[Constantopoulos95] Constantopoulos P., Jarke M., Mylopoulos J., Vassiliou Y.: The Software Information Base -- A Server for Reuse. VLDB Journal 5(1), 1-42, 1995.

[Easterbrook96] Easterbrook S,: Learning from Inconsistency; Proc. of 8th Intl. Workshop on Software Specification and Design, Velen, Germany, March 1996.

[Gotel94] Gotel O., Finkelstein A.W.: Modelling the Contribution Structure Underlying Reqirements; Proc. of the first International Workshop on Requirements Engineering, 71-88, Utrecht, The Netherlands, June 1994.

[Higa95] Higa K., Ma P.C. Smith M.A.: Defining Business Constraints in Relational Databases Using a Semantic Data Model. Proc. 3rd Intl. Conf. Intl. Society for Decision Support Systems, Hong Kong, June 1995.

[Hong95] Hong S., Brinkkemper S., Harmsen F.: Object-Oriented Method Components for Situation-Specifec IS Development, Proc. 5th Workshop on Information Technologies and Systems (WITS'95), 164-173, Dec. 1995.

[Jacobson92] Jacobson I, et al.: Object-Oriented Software Engineering A Use Case Driven Approach; Addison Wesley, 1992.

[Jarke95] Jarke M., Gallersdörfer R., Jeusfeld M.A., Staudt M., Eherer S.: ConceptBase - a Deductive Object Manager for Meta Data Management; *Journal of Intelligent Information Systems*, 4(2), 167-192, 1995.

[Jarke96] Jarke M., Gebhardt M., Jacobs S., Nissen H.W.: Conflict Analysis Across Heterogeneous Viewpoints: Formalization and Visualization; Proc. 29th Hawaii Intl. Conf. on System Sciences, 199-208, 1996

[Kotonya92] Kotonya G., Sommerville I.: Viewpoints for requirements definition; *Software Engineering Journal*,375-387, November 1992.

[Leite91] Leite J.C.S.P., Freeman P., A.: Requirements validation through viewpoint resolution; *IEEE TSE* 12(12), 1253-1269, December 1991.

[Maiden95] Maiden N., Assenova P., Constantopoulos P., Jarke M., Johannesson P., Nissen H.W., Spanoudakis G., Sutcliffe A.: Computational Mechanisms for Distributed Requirements Engineering; Proc. of 7th SEKE, Rockville, Maryland, USA, June 1995.

[Motschnig-Pitrik95] Motschnig-Pitrik R.: An Integrating View on the Viewing Abatraction: Contexts and Perspectives in Software Denelopment, AI, and Databases; *Journal of Systems Integration*, Kluwer, 5 (1), 23-60, April 1995.

[Motschnig-Pitrik96] Motschnig-Pitrik R.: Analysing the Notions of Attribute, Aggregate, Part, and Member in Data/Knowledge Modelling; *The Journal of Systems and Software*, 33,113-122, 1996.

[Mullery79] Mullery G.P.: CORE - a method for controlled requirements specification; Proc. of the 4th International Conference on Software Engineering, 126-135, Munich, Germany, 1979.

[Mylopoulos90] Mylopoulos, J., Borgida, A., Jarke, M. and Koubarakis, M., Telos: Representing Knowledge About Information Systems, *ACM Transactions on Information Systems*, 8(4), 325-362, Oct. 1990.

[Mylopoulos95] Mylopoulos J., Motschnig-Pitrik R.: Partitioning Information Bases with Contexts; Proc. 3rd International Conference on Cooperative Information Systems, CoopIS'95, 44-54, Vienna, May 1995.

[Nissen96] Nissen H.W., Jeusfeld M., Jarke M., Zemanek G.V., Huber H.: Managing Multiple Requirements Perspectives with Metamodels. *IEEE Software*, 37-48, March 1996.

[Nissen97] Nissen H.W. Separation and Resolution of Multiple Perspectives in Conceptual Modeling. Doctoral thesis (in German), RWTH Aachen, 1997.

[Nuseibeh94] Nuseibeh B., Kramer J., Finkelstein A.W.: A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification; *IEEE TSE* 20(10), 760-773, October 1994.

[Pohl94] Pohl K.: The three dimensions of requirements engineering: a framework and its application; *Information Systems*, 19(3), 1994.

[Ross77] Ross D., T.: Structured analysis: a language for communicating ideas; *IEEE TSE* 3(1), 16-34, 1977.

[Rumbaugh91] Rumbaugh J., Blaha M., Premerlani W., Eddy F., Lorensen W.,: Object-Oriented Modeling and Design, Prentice Hall, Englewood Cliffs NJ, 1991.

[Scheer94] Scheer A.-W.: Business Process Engineering; Springer-Verlag, 1994.

[Shaw84] Shaw M.: The Impact of Modelling and Abstraction Concerns on Modern Programming Languages; in: Brodie M., Mylopoulos J., Schmidt J.W. (eds.), On Conceptual Modeling, Springer-Verlag, 1984.