Restructuring Requirements Specifications for Inconsistency Analysis: A Case Study

Alessandra Russo Bashar Nuseibeh
Jeff Kramer

Department of Computing, Imperial College
180 Queen's Gate, London SW7 2BZ, U.K.
{ar3, ban, jk}@doc.ic.ac.uk

Abstract

This paper describes our experiences in *restructuring* multi-perspective requirements specifications in order to identify and analyse possible or existing inconsistencies. A partial, heterogeneous and reasonably large requirements specification from a NASA project was analysed and decomposed into a structure of "viewpoints", where each viewpoint encapsulates partial requirements of some system components described in the specification. Relationships between viewpoints were identified which included not only the interactions explicitly stated in the requirements but also some *implicit* and *potentially problematic* inter-dependencies. The entire restructuring process and a first informal analysis of the resulting relationships enabled the detection of (potential) inconsistencies and the definition of some interesting *domain-dependent* consistency rules. We believe that this restructuring into viewpoints also facilitates requirements understanding through partitioning, and requirements maintenance and evolution through explicit identification of the inter-viewpoint relationships.

1 Introduction

The requirements engineering process of large and complex systems often involves the participation of many developers who operate according to their specific skills, experience and knowledge. This is particularly the case for the elicitation and development of requirements specifications. Multiple participants are normally involved in specifying different partial requirements of the same underlying system. These are inevitably described using various development methods and diverse representation schemes, reflecting the different knowledge and *perspectives* that the developers have about the underlying domain. This heterogeneity of the resulting requirements specifications makes them less amenable to analysis and change management. Relationships between the various fragments such as overlaps, intersections and inter-dependencies are difficult to identify. Consequently, activities like consistency checking and the analysis and handling of inconsistencies become much harder. To address this problem we need an analysis, *a priori*, of the relationships between the methods and representation styles used to develop the specification fragments, as well as, relationships between the domain-specific terms and concepts (e.g., identifying different terms which denote the same object). Given that it is not always feasible to develop multi-perspective specifications with a structure amenable to analysis and management, an approach that facilitate *restructuring* of *existing* specifications seems appropriate.

This paper describes our practical experience in restructuring a partial, reasonably large, multi-perspective requirements specification of a NASA project called *International Space Station* (ISS). We deploy the Viewpoints approach [5, 8] as a restructuring tool. This approach facilitates an explicit separation of the perspectives of the different developers and their representation by "ViewPoints"[1], allowing the use of different development methods and representation schemes. Inter-viewpoint rules can be defined which enable not only a structural integration of the different requirements specification fragments, but also provide a means for consistency checking and inconsistency analysis. This paper illustrates that these same features also make the Viewpoints framework useful for restructuring existing requirements specifications to facilitate impact analysis and change management.

Section 2 describes, with some examples, the case study we used. This is an informal, but structured, partial requirements specification of an integrated hardware and software system (C&DH), which is part of a much wider system for handling the operation of an earth-orbiting space station. We focus our attention on a particular function of the C&DH, namely, the "Fault, Detection, Isolation and Recovery" (FDIR) function. In Section 3 we illustrate how this informal requirements specification has been restructured into viewpoints, discussing some of the difficulties and problems encountered in this process. We also discuss the advantages of the restructuring process for detecting inconsistencies. The resulting structured viewpoints' representation of the requirements specification provided easier access to the document, thus allowing a better understanding of the underlying domain. Furthermore, the identification and representation of *implicit relationships* between various parts of the specifications facilitated the detection of some forms of inconsistencies and highlighted interesting problems, not necessarily related to the specific domain. This is illustrated in Section 4, where some examples of inconsistencies are also given. The paper concludes with a discussion on extensions of this work and its suitability for managing the evolution of large multi-perspective requirements specifications. We strongly believe that our investigation provides a useful testbed for investigations into the detection of, and response to, *evolutionary changes* in multi-perspective requirements specifications.

[1] In this paper, we write the term "viewpoint" in lower case throughout, even though we are still referring to our particular form of "ViewPoints" described in earlier work [5, 8].

2 The Case Study

The case study under consideration is a partial, multi-perspective requirements specification of a NASA project called *International Space Station* (ISS). The project is an international co-operation programme for the construction and use of a space station orbiting earth. The space station construction process deploys earth-to-orbit vehicles (e.g., the Space Shuttle and Ariane) provided by various international partners to launch flight elements, systems for assembly, as well as crew into orbit. Flight crews operate from the Shuttle to assemble the Station using a combination of robotics and "extravehicular activities". During the building process the station is periodically reboosted to a higher orbit to compensate for orbital decay. Engineering and science disciplines such as biological and life sciences, microgravity, and technology research are expected to benefit significantly from this first part of the programme. The mission of the station is to facilitate scientific experiments in a microgravity environment, exploiting international earth-orbiting research facilities, throughout its lifetime.

The space station system includes "orbit" and "ground" facilities which are linked to each other via data communication. This facilitates the *control* and monitoring of the station operations by a ground team of controllers. The requirements specifications of the space station system refers to both flight and ground hardware/software components.

One of the main functions of the space station software is the *Command and Control* (C&C) function. This is responsible for the station operations by monitoring its functions, overriding them when necessary, performing fault detection, and issuing commands to control the main station modes (e.g., docking with a Shuttle, re-boosting to a higher orbit, maintaining microgravity for science experiments). The C&C function is carried out as part of the ISS Command and Data Handling (C&DH) system. The C&DH system is comprised of a set of Computer Software Configuration Items (CSCIs) installed in a hierarchically organised set of processors (Multiplexer/Demultiplexers - MDMs) which communicate among themselves via MIL-STD-1553 buses. The main flight processor responsible for the C&C function is denoted as the "C&C MDM". The CSCI within the C&C MDM is the Command and Control Software (CCS) CSCI. The C&C MDM is located at the top of the hierarchy and is the comtroller of the 1553 buses to the MDMs in the hierarchy tier directly below (therefore, referred to as the Bus Controller - BC). The MDMs (along with other hardware) at this next tier are designated as Remote Terminals (RTs) on the buses.

The case study under consideration refers to two partial requirements specifications for the CCS, namely, the *Control Bus MDM Management* (4 pages) and the *1553 Bus Failure, Detection, Isolation and Recovery* (20 pages). The former is a requirements document describing the management of MDMs connected to the control bus (one of these MDMs is also the CCS MDM). This management includes initialisation, shutdown and *failure recovery.* The latter document describes the 1553 Bus Failure, Detection, Isolation and Recovery, also denoted by FDIR, which specifies the management of control buses[2]. Such management consists partly of checking the control buses periodically, for the identification of the control bus in use, and for *failure detection* of processor RTs connected to such a control bus.

These specifications are two independent documents, extracted from the whole requirements description of the space station system. The section numbers used in the documents show a discontinuity which indicates the existence of a large intermediate section of the requirements specifications, not included in our investigation. Nevertheless, after a first analysis, based on comparing some of the tables referred to by these specifications, we have identified a link between the two documents, namely that some of the RTs on which the FDIR performs failure detection are the MDM processors on which the Control Bus MDM Management performs its failure recovery. Using this observation, we have been able to identify, throughout the restructuring process, various relationships between these two documents. The parts which appear to be more related to each other are the *failure detection* and the *failure recovery* components, which are described respectively in the FDIR and in the Control Bus MDM Management. A more detailed description of our restructuring approach and of the types of relationships we identified are presented in the following section.

3 Restructuring Requirements Specifications Using Viewpoints

This section illustrates our approach to restructuring parts of the existing requirements specifications for the case study described above.

3.1 Our approach

The first important question which arises in investigations of this kind is how to decompose a given existing requirements specification. Various criteria could be adopted, such as to define a viewpoint for each function and subfunction described in the specifications, to choose partitions on the basis of the type of representation styles used in the specifications, or simply to follow the existing structure of the requirements specification documents. The first solution is not always easy to adopt since specification fragments like those under consideration do not include complete lists of the related functions and sub-functions, and their identification would require a careful analysis, *a priori*, of the specifications. Moreover, the second solution of performing the decomposition on the basis of the representation styles does not guarantee a sufficiently fine-granularity. In our case study for example, most of the requirements were described in natural language text. Thus, the third solution of decomposing the requirements specifications according to the existing documents' structure appeared to be the

[2]The space station system could essentially be regarded as one big distributed system with various communication and control buses, both ground and orbit.

most direct and immediate approach. Indeed we adopted this approach because it also guarantees a decomposition which is *complete* with respect to the given specifications.

The first viewpoint we defined provides a hierarchical tree structure representation of the two requirements documents under consideration. In this structure the root node corresponds to the global system CCS CSCI and from it two subtrees are defined which correspond, to the two requirements specifications; i.e. the Control Bus MDM Management and the 1553 Bus FDIR. In each of these subtrees the parenthood relation of the nodes corresponds to the sections and sub-sections structure of the relative documents. An example of part of this tree structure is given in Figure 1.

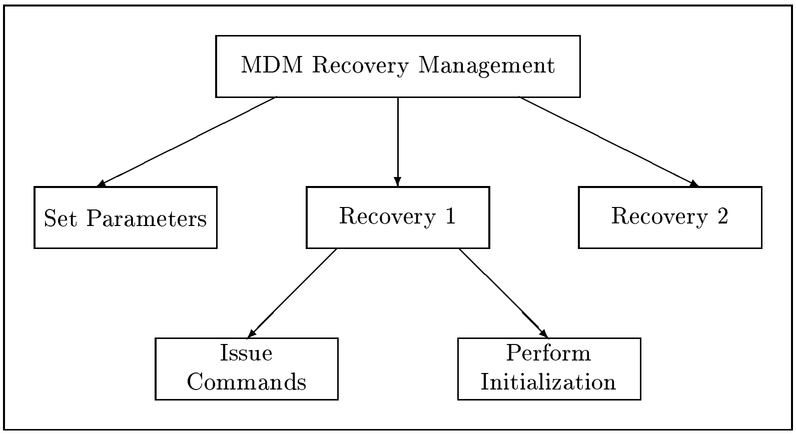

Figure 1: Part of the HT viewpoint.

A more detailed analysis of the two requirements specification documents subsequently showed us that the sections/sub-sections division provides a useful first high-level functional decomposition. Each leaf of the tree was further decomposed into sub-nodes, each corresponding individually to the sub-functions described in the section related to that leaf node. We called the resulting viewpoint a *hierarchical tree* (HT) viewpoint, where hierarchical tree is the name of the representation style we adopted (or Viewpoints *template* in the terminology of the Viewpoints framework [5]).

We then used another two viewpoint representation styles which we call “Data-Flow Tables” (DFT) and “Input-Output Flows” (IOF) to express more detailed requirements specifications for each node in the above tree viewpoint. The DFT allows the representation of processes, and the input and output data flows between these processes and the external environment. The IOF is similar to the tabular collection representation style used in the CORE method [7], which also includes the “sources” and “destinations” of the inputs and output flows, respectively.

To achieve a reasonably fine level of granularity in our decomposition, we allow for *primitive* and *non-primitive* processes in a DFT. This is because some of the identified processes have specifications more complex then others (based on the number of – possibly alternative – actions and additional conditions expressed in the processes' requirements specifications). We call primitive processes those that are specified as a single action to be taken whenever certain data flows (inputs) reach the process, and *non-primitive* processes, those who are specified in terms of more than one action.

For each parent node (with the exception of the root node) in the HT viewpoint, we instantiated a viewpoint from the DFT template. In this viewpoint we specified the processes (or subfunctions) stated in the node's requirements specification and the data flow information between these processes and the external environment. An example is given in Figure 2 for the node 'MDM Recovery Management' that appears in the tree given in Figure 1.

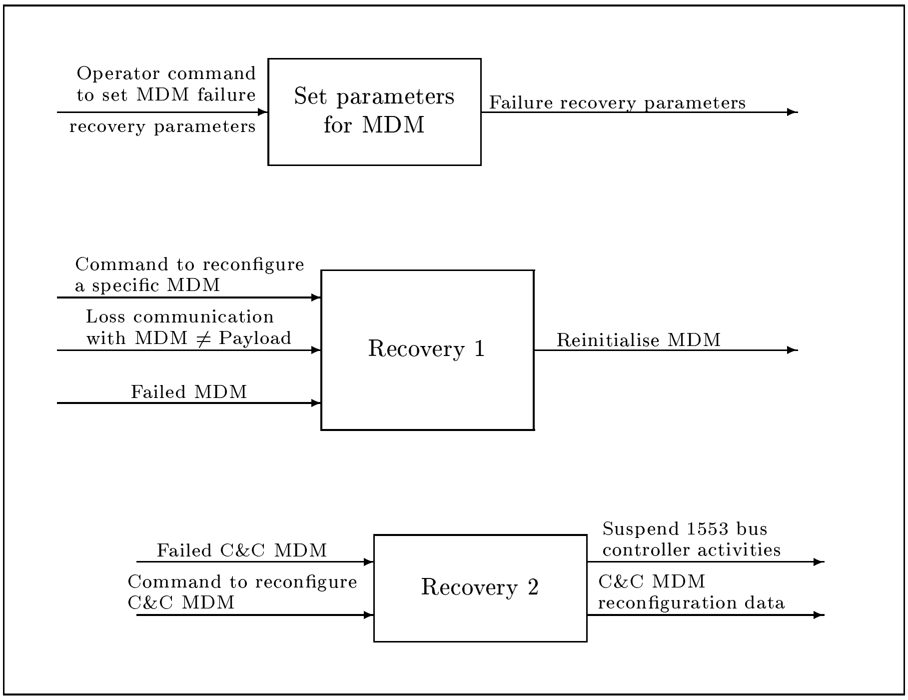

Figure 2: A DFT instantiation for MDM Recovery Management.

The main role of the MDM Recovery Management is to handle the recovery of MDMs upon the receipt of a failure. Different actions are taken according to the type of MDMs. The process Recovery 1 describes the actions to be taken when an MDM different from the main C&C MDM has failed, whereas the process Recovery 2 describes the response of the MDM Recovery Management upon the receipt of the C&C MDM failure. Recovery 1 is non-primitive because it includes additional controls to be performed before the recovery management is activated.

We also defined two general inter-viewpoint rules between the two templates. The first one belongs to the HT template and it is of the following form:

> *For all X, if X is a parent node then there exists a viewpoint instantiation from the DFT template with domain X.*[3]

The second general inter-viewpoint rule belongs instead to the DFT template and it is defined as follows:

> *For all X, if X is a process then, if there exists a Y viewpoint instantiation from the HT template, then there exists in Y a sub-node labelled with X whose parent node is labelled with the same domain of X.*

These rules allow syntactic completeness checking of the restructured representation.

For each leaf of the HT viewpoint and for each non-primitive process of a DFT viewpoint, we instantiated a viewpoint from the template IOF (for example, we instantiated an IOF viewpoint for the non-primitive process Recovery 1 showb in Figure 2). The inputs in such viewpoint instantiations can be either the inputs which appear in the DFT viewpoint instantiation for the parent node or are additional inputs specific to the particular action described in the sub-node. To capture this structural property we defined appropriate inter-viewpoints rules between the templates HT and IOF, and between the templates DFT and IOF, which are again rules for completeness checking.

3.2 Defining additional relationships

Throughout the restructuring process we were able to identify several other relationships between viewpoints. We classified them into three different types. The first includes relationships such as those described in the previous section. They allow us to check whether our representation of the requirements specifications is complete. Relationships of the second type express the 'interface communication' between viewpoints. A typical example is a relationship between two viewpoints X and Y which states that some of the output of X is used as input data in the viewpoint Y. Relationships of this kind have been widely discussed in [2, 8], where, for the specific case of the IOF template, they are defined as matching the output and destination of one viewpoint X with the input of another viewpoint Y whose domain is equal to that destination in X. However, in our case study most of the tabular (IOF) viewpoints have empty sources

[3]The term "domain" is used in the Viewpoints framework [5] to refer to a Viewpoint's area of concern.

and destinations because of missing information in the requirements specifications. We addressed this problem by considering pattern-matching between the viewpoints' contexts. In this way, we were able to identify a number of such relationships, and making them explicit allowed us to complete some of the viewpoints' *missing information* about their source and destination. This is one of the advantages of our restructuring process of existing requirements specifications.

The third type of relationship are domain-specific relationships. These are the most difficult to identify because they often depend on features of particular systems, and are often left implicit in requirements specifications. Because they are domain-dependent, it is very difficult to draw any general rule which could help us to deduce such relationships. What we need instead is a kind of "abduction" process, to abduce (hypothesise) domain-specific relationships between viewpoints which, when added to the rest of the requirements specifications, still allow us to infer the same information about the underlying system. We were able to identify one such relationship between two viewpoints – the RT Failure Detection and the MDM Recovery Management (see Figure 1). As mentioned above, the latter describes the CCS CSCI behaviour response to the indication of an MDM failure. If the MDM is different from the C&C MDM then the recovery response, handled by the sub-node Recovery 1 of MDM Recovery Management (see Figure 1), initialises the MDM to its backup. If the failed MDM is the C&C MDM then the recovery response, which is now handled by the Recovery 2 (see Figure 1) sub-node, is the CCS CSCI suspends any control activity on the 1553 buses and reinitialises itself. On the other hand, the viewpoint RT Failure Detection describes the CCS CSCI response to its detection of a failure on an RT (recall that RTs also include MDMs). One of its responses is to suspend its failure checking activity on the 1553 buses for a certain period of time (this is left undefined, "TBD", in the specifications). Considering that the RT Failure Detection is already related to the MDM Recovery Management whenever the failed RT is an MDM – RT Failure Detection's output of identifying an RT failure is used as an input data by the MDM Recovery Management – we have defined that in these circumstances the time TBD of the RT Failure Detection is related to the two times (of the MDM Recovery Management) spent by the CCS CSCI to either reinitialise an MDM or to reinitialise itself.

The definition and analysis of all these types of relationships has allowed us to identify in the given requirements specifications some existing inconsistencies and open problems. A discussion and an informal evaluation of our approach is presented in the following section.

4 Discussion and Evaluation

In the previous sections, we suggested that the application of our restructuring approach facilitated the identification of some inconsistencies and potential problems in the requirements specifications. We now discuss particular examples. The main inconsistencies we discovered were within the requirements specification of the part called "Bus Channel Management" of the 1553 Bus FDIR document.

The 1553 Bus FDIR specification includes a flowchart description of the general behaviour of the Bus Channel Management. We represented it as a separate viewpoint, deploying a representation style based on state transition diagrams (STD). This viewpoint describes the various sets of conditions which trigger the different Bus Channel Management's actions. In the HT viewpoint the Bus Channel Management is also a parent node with various sub-nodes and related leaves. This is shown in Figure 3.

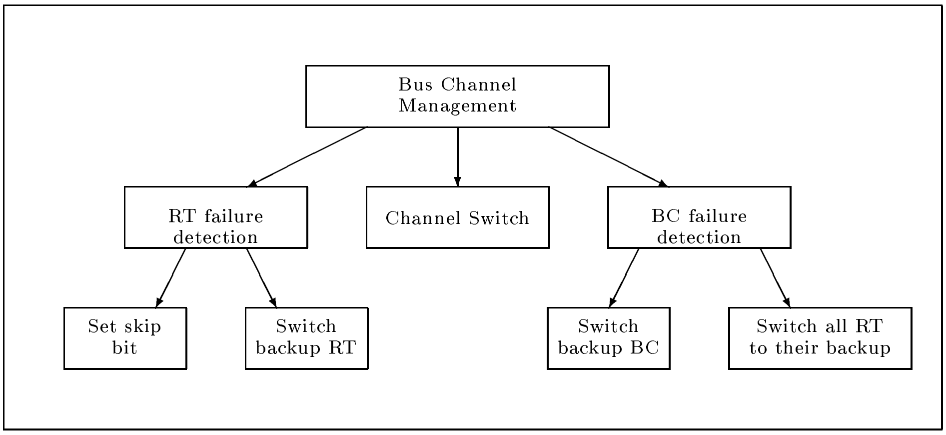

Figure 3: Part of HT viewpoint for the Bus Channel Management.

The leaves of this particular sub-tree correspond to some of the actions described in the viewpoint instantiated from the STD template. For each of these leaves we instantiated an IOF viewpoint which expresses the text description of the requirements specifications for the Bus Channel Management's actions. In these viewpoints the inputs to a specific action correspond to the conditions stated in the STD viewpoint for that action. This restructuring allowed us to identify domain-specific inter-viewpoint relationships between the STD viewpoint and these viewpoints. One such relationship states that in each of these IOF viewpoints, for each set of inputs, there must to exist a set of conditions in the STD viewpoint which corresponds to the action represented in the particular IOF viewpoint. Applying this relationship to some of the IOF viewpoints, we were been able to *deduce inconsistencies* between them and the STD viewpoint; i.e. the IOFs include some inputs which are not specified in the STD viewpoint. This inconsistency shows that either the flowchart of the Bus Channel Management is not correct or the text specifications of the individual actions are wrong. Consulting NASA, we then learned that this same inconsistency was also identified by the Independent Verification and Validation (IV&V) team, and that the likely source of the inconsistency most problably lies in the flowchart, since it was produced at a very early stage of the requirements specification process – before the text descriptions were subsequently updated and changed.

This is a typical example where we believe that our restructuring approach provides at least two advantages across the whole process of requirements specifications development. Firstly, the restructuring approach is a helpful tool for identifying inconsistencies. This was supported by the fact that it identified the same inconsistencies identified by other approaches such as the IV&V process deployed by NASA [1]. Secondly, the restructuring approach provides a useful tool that facilitates consistent updating and extension of existing requirements specifications. In the above example, inter-viewpoint relationships can guide the requirement developers in checking the STD viewpoint whenever a change is made to a related IOF viewpoint. Of course, in order to make this practically feasible, it is important to manage the traceability links between the restructured representation and the original requirements specifications. This is a challenging open problem identified by our investigation.

Our restructuring approach also has the advantage of facilitating the identification of *potential* problems in existing requirement specifications. We were able to identify one in particular in our case study. In Section 3.2 we described a domain-specific relationship between the viewpoints RT Failure Detection and MDM Recovery Management. This is that the TBD time of the CCS CSCI in the RT Failure Detection corresponds to either the time used by CCS CSCI in the MDM Recovery Management to reinitialise the failed MDM or to the time used by the CCS CSCI in the MDM Recovery Management to reinitialise itself. At the moment, these specifications are consistent. However, the potential problem is that future instantiations of the TBD time might cause inconsistencies. For instance, it could be the case that the TBD time is instantiated with a value which is less than the time spent by the CCS CSCI to reinitialise itself.

This example raises the following question. Is it necessary to specify consistency rules between parameters whose values have not been defined yet? Our approach suggests that this is indeed useful. We try to include all domain-dependent relationships whenever these are identifiable and definable, in order to detect inconsistencies that may be introduced later in the development process when the specification changes or evolves. The restructuring approach facilitates the identification of these relationships. In the example above, it suggests for instance as a possible consistency relationship that any instantiation of the TBD in the RT Failure Detection should be greater than the times specified in the MDM Recovery Management.

Restructuring also allows the identification of missing information in existing requirements specifications; it highlights relationships, cross-references and inter-dependencies left implicit in the specifications to be explicitly expressed; and finally, it provides a tool for detection of overlaps and intersections between various fragments of the specification. All these advantages contribute significantly towards addressing the more general problem of detecting and handling inconsistencies [3, 4, 6], not only at a particular stage of the requirements development, but also during its modifications and extensions. We believe that this apprach complemented with good traceability tools that navigate between the restructured representation and the actual requirements documents, provides a powerful means of managing change, and analysing its impact. We are currently developing automated tools to support distributed development of viewpoints and analysis of inconsistencies, and we have preliminary prototypes of each written in Java and Prolog respectively.

Acknowledgment

We would like to thank Steve Easterbrook (WVU) for providing us with the case study, and for his patient and helpful answers to our endless clarification questions. Thanks also to John Hinkle (NASA) for his feedback which helped to improve the paper. The research described in this paper was funded by the UK EPSRC project VOILA (GR J15483). We are also grateful for financial support from NASA's IV&V Facility, West Virginia, USA.

References

[1] S. Easterbrook and J. Callahan. Formal Methods for V&V of Partial Specifications: An Experience Report, In Proceedings of 3rd International Symposium on Requirements Engineering (RE'97), 160-168, IEEE Computer Society Press.

[2] S. Easterbrook, A. Finkelstein, J Kramer and B. Nuseibeh. Co-ordinating Distributed ViewPoints: The anatomy of a consistency check. *Concurrent Engineering: Research and Applications*, 2(3): 209-222, CERA Institute West Bloomfield, USA.

[3] S. Easterbrook and B. Nuseibeh, Using ViewPoints for Inconsistency Management. *Software Engineering Journal*, 11(1): 31-43, January 1996, BCS/IEE Press.

[4] A. Finkelstein and D. Gabbay and A. Hunter and J. Kramer and B. Nuseibeh. Inconsistency Handling in Multiperspective Specifications. *Transactions on Software Engineering*, 20(8): 569-578, August 1994, IEEE Computer Society Press.

[5] A. Finkelstein and J. Kramer and B. Nuseibeh and L. Finkelstein and M. Goedicke. Viewpoints: A Framework for Integrating Multiple Perspectives in System Development. *International Journal of Software Engineering and Knowledge Engineering*, 2(1):31-58, March 1992, World Scientific Publishing Co.

[6] A. Hunter and B. Nuseibeh. Analysing Inconsistent Specifications. In Proceedings of 3rd International Symposium on Requirements Engineering (RE'97), 78-86, IEEE Computer Society Press.

[7] G. Mullery. CORE - a method for controlled requirements expression. In Proceedings of 4th International Conference on Software Engineering (ICSE-4), 126-135, IEEE Computer Society Press.

[8] B. Nuseibeh, J. Kramer and A. Finkelstein. A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification. *Transactions on Software Engineering*, 20(10): 760-773, October 1994, IEEE Computer Society Press.