ViewPoints: meaningful relationships are difficult!

Bashar Nuseibeh
Department of Computing
The Open University
Walton Hall
Milton Keynes MK7 6AA, UK
B.A.Nuseibeh@open.ac.uk

Jeff Kramer
Department of Computing
Imperial College
180 Queen's Gate
London SW7 2BZ, UK
j.kramer@imperial.ac.uk

Anthony Finkelstein
Department of Computer Science
University College London
Gower Street
London WC1E 6BT, UK
a.finkelstein@cs.ucl.ac.uk

Abstract

The development of complex systems invariably involves many stakeholders who have different perspectives on the problem they are addressing, the system being developed, and the process by which it is being developed. The ViewPoints framework was devised to provide an organisational framework in which these different perspectives, and their relationships, could be explicitly represented and analysed. The framework acknowledges the inevitability of multiple inconsistent views, promotes separation of concerns, and encourages decentralised specification while providing support for integration through relationships and composition. In this paper, we reflect on the ViewPoints framework, current work, and future research directions.

1 Multi-Perspective Software Development

Any sizeable software development effort will involve a variety of stakeholders with different aspects of concern – or *perspectives* – that overlap, complement, or contradict each other. These stakeholders express their concerns using a variety of representations, and follow different processes to deploy those representations. This makes determining the relationships between different perspectives particularly difficult, yet crucial for checking consistency and managing inconsistency between those perspectives.

Such *multi-perspective software development* has motivated our research for over a decade. It embodies a number of characteristics that continue to lie at the heart of organising the software development activity. These characteristics include the need for separation of concerns during software development, the inevitability of having multiple inconsistent views of processes and products of software development, and the need to reason analytically over multiple views in order to understand the properties and consequences of a multi-perspective specification.

Much of this research has focused on requirements engineering (RE) activities within the software engineering life cycle. RE is prototypical of multi-perspective software development. Multiple stakeholders hold different views of the problems to be addressed, which in turn determine the different requirements of the system to be developed. These requirements are often inconsistent, but acceptably so, during some part of the RE process when requirements are still being elicited, negotiated, and prioritised.

This paper reflects briefly on the *ViewPoints framework* [10, 33], which we developed to organise and manage multi-perspective software development. The paper reviews the basic components of the framework, their *raison d'être*, and the way in which we have deployed them over the last ten years. The paper describes the use of explicit relationships between ViewPoints to manage multi-perspective software development, and reports on our most recent incarnation of the ViewPoints framework in the form of the xlinkit toolkit. The paper then speculates about the need for more meaningful relationships between multiple ViewPoints in order to reason more effectively about structured collections, or configurations, of such ViewPoints.

2 The ViewPoints Framework

The ViewPoints framework provided an infrastructure for capturing and organising software development knowledge. Our notion of a ViewPoint was of an object encapsulating cross-cutting and partial knowledge about notation, process, and domain of discourse, from the perspective of a particular stakeholder, or group of stakeholders, in the development process.

A system description – a specification – thus comprises a structured collection of loosely coupled, locally managed, distributable ViewPoints, with explicit relationships between them to represent their overlaps. These overlaps are the focal point for consistency checking between ViewPoints, and subsequent inconsistency management, should ViewPoint relationships be found not to hold.

2.1 Key principles

ViewPoints re-iterate a number of key, well-documented principles of software engineering, and surface a number of new ones.

Fundamentally, ViewPoints organise software development knowledge based on *separation of concerns*. A ViewPoint, expresses the concerns of a particular stakeholder, such as a development participant or a representative of an area of concern captured by that ViewPoint. Thus, a ViewPoint may represent an area of concern within a project, a product, or a process, or may simply present a particular

perspective expressed in a particular notation. The choice of dimensions of concern along which to create ViewPoints may be the result of experience, the nature of the problem at hand, or simply organisational exigencies.

The next, but related, key principle is that of *heterogeneity of representations*. ViewPoints deliberately allow different perspectives to be represented using different notations, based on the observation that stakeholders will present their perspectives using different notations, depending on their preferences, needs, or circumstances. Of course, relating different perspectives will inevitably require some expression of the relationships between the different notations, and ViewPoints provide a framework for precisely such an expression.

Using the ViewPoints framework leads directly to the notion of distributable ViewPoints – that is, ViewPoints that can be distributed physically or logically. The key principle upon which this relies is *decentralisation* – the idea that the knowledge encapsulated within each ViewPoint is elicited locally, developed locally, and managed locally. Such decentralisation explicitly moves away from any notion of a monolithic system specification that can be checked or managed globally, towards collections of loosely coupled partial descriptions.

The consequence of heterogeneity, decentralisation, and partiality is that relationships between ViewPoints are crucial for achieving *integration* in its many forms. In the ViewPoints framework, integration is often methodological; that is, it is about meaningful linking of processes and notations in order to achieve *coordination* and *composition*, respectively [32]. Thus, methods may be *engineered* by relating different notations and prescribing when and how they are used. Also, methods may be *deployed*, leading to many ViewPoints linked by instantiations of relationships defined during method engineering.

Another consequence of a specification comprised of many different ViewPoints is the inevitable inconsistencies that arise between those ViewPoints. The approach adopted by the ViewPoints framework is one of *living with inconsistency*, in order to capture the diverse perspectives of stakeholders. Consistency is defined by *inter-ViewPoint rules* that express the (static semantic) relationships between ViewPoints [6, 34]. *Inconsistency management* is then the process of handling the potential plethora of rules and dealing with the ViewPoints when the desired relationships between them do not hold [7, 12, 35, 37].

2.2 Why ViewPoints are still relevant

The principles upon which ViewPoints are based continue to be relevant in the context of software development today. The aspect-oriented software development (AOSD) community has focused in recent years on precisely the kind of "cross-cutting" concerns investigated within the ViewPoints framework. This community has focused mostly on concerns in programming code, however, there is now increasing recognition that these concerns may also manifest themselves in requirements and designs, and so consideration of 'early aspects' may be necessary [4, 40].

The ViewPoints framework also provided an opportunity for method engineering – the customisation of methods by mixing notations [36], processes and guidelines for their deployment. During such method engineering, different types ("templates") of ViewPoints can be assembled together, and related, leading to multiple, heterogeneous ViewPoints when the methods are deployed (or in ViewPoints' terminology, when ViewPoint templates are instantiated). While method engineering as a research discipline, has not prospered it is in fact widely practiced. The Unified Modelling Language (UML) with its multiple, heterogeneous modelling notations, user-defined profiles, extension mechanisms, and variety of processes in which it is used, may be seen as an example of the move towards choosing the most appropriate method for the problem in hand [1]. Expressing and checking the relationships between the various models remains a challenge.

Decentralisation was at the heart of the ViewPoints framework. Many of the issues identified are of direct relevance for the provision of tool support to distributed software engineering teams working cooperatively and asynchronously [21, 30]. The consequences of supporting multiple views, namely, the need to live with and manage inconsistency between those views are of particular interest [3, 9, 15, 16].

2.3 A critical analysis

The software engineering community in general, and the RE community in particular, appear to have accepted the need to articulate and manage multiple views in the software development process. There are a variety of concerns along which these views may be separated, including aspects, actors, representation schemes, or processes. Dividing large monolithic specifications into many smaller partial specifications is one way of managing such specifications.

The ViewPoints framework articulated a research agenda in which the above ideas featured. It also, through cases studies and demonstrator tool support, presented a vision of multi-perspective software development in which methods can be engineered, many partial, inconsistent specifications created, and consistency relationships between them checked. The research agenda changed over time and assiduous reading of our papers will reveal different and, ironically, not wholly consistent variants of the framework.

What our early demonstrators did not achieve were truly heterogeneous representations. It is also fair to accuse our early work of providing a reference model without a plausible distributed implementation to support it.

Distribution has been largely addressed in our more recent

work on xlinkit, a toolkit for checking documents distributed on the web, described in the next section. Heterogeneity has posed somewhat more problems. The ViewPoints framework made great polemical play of heterogeneity. The coordinated use of multiple modelling notations is now common and widely accepted as the right approach to specification. It is not at all clear that our advocacy of heterogeneity in representing mappings between modelling representations and consistency relationships between models using multiple these notations was the right approach. Certainly, our later work [6, 12, 18, 27] has adopted a single meta-language for expressing the relationships between multiple views (as UML does in its adoption of a single meta-model and framework of constraints).

The single language is crucial for performing meaningful analytical reasoning. In much of our work we have adopted predicate logic as an underlying representation upon which to perform reasoning [12]. In some instances we found it necessary to adapt classical logic in order to perform reasoning in the presence of inconsistency [18]. In other cases, we used particular forms of classical logic in order to reason about the changing specifications [14, 41]. Most recently, in the context of xlinkit, we have based our techniques for repairing inconsistencies between specifications on first order predicate logic with some significant practical success [29].

3 xlinkit

xlinkit is a direct successor to much of the work on ViewPoints. It provides a partial realisation and a demonstration that many of the core ideas of the framework were sound. It has also, as implementations are wont to do, surfaced a wide range of new issues, to which we had paid scant attention in our earlier work.

xlinkit, a lightweight application service that provides rule-based link generation and checks the consistency of distributed web content. A full description can be found in [27, 28, 29] and online demonstrations, tutorials and other materials can be found at `http://www.xlinkit.com`.

The operation of xlinkit is quite simple. It is given a set of distributed XML resources and a set of potentially distributed rules that relate the content of those resources. The rules express consistency constraints across the resource types. xlinkit returns a set of XLinks, in the form of a linkbase, that support navigation between elements of the XML resources. Here XML is used as the common interchange language to overcome heterogeneity.

The xlinkit rule language is first-order logic (the concrete syntax being XML). The xlinkit 'check engine' provides a novel linking semantics to this language that returns hyperlinks between inconsistent elements instead of boolean values.

The hyperlinks provide very precise diagnostics that we believe are essential for inconsistency management. xlinkit has a reporting framework 'Pulitzer' that uses these diagnostics to provide informative reporting to resource, aka viewpoint, owners.

Recent work, reported at this conference, has concentrated on 'repair'. Using the rules, by static analysis, we can identify a set of repairs that can be used to restore consistency. These repairs, or at any rate a sensible subset of them, as determined by an appropriate authority, can be used in conjunction with our diagnostic reporting to support inconsistency management.

xlinkit leverages standard Internet technologies. It supports wide-scale document distribution and can support multiple deployment models. It has a strong formal foundation. We have completed a number of large-scale evaluations, including a study that checks UML models, Java code, EJB deployment information and UML profile information. This and similar studies give us confidence in xlinkit and demonstrate that we have gone some way towards making ViewPoints a reality. xlinkit is now being commercialised by Systemwire, a research spinout company.

Experience with xlinkit has brought home to us the challenges of scalability, in terms of the size of documents, the number of rules and the complexity of those rules. We have implemented an incremental checker, a distributed checker, smart caching using an XML repository and an ultra-high performance replicated checker. We have also devoted a large amount of effort to optimisation with a view to ensuring that checking and rechecking can be done on large document collections as and when necessary. Ensuring usability, good interfaces and meaningful error messages has also occupied much time. These are not peripheral issues, they have been essential in order to be able to complete the case studies, and attract the external users, that validate our work.

There is a substantial future research agenda for xlinkit. One of the key items on this agenda is the relationship between checking and workflow. The ViewPoints framework envisaged that checks would be triggered by the enaction of a process model. This process model would identify which checks should be applied when and the results of the checks could then be used to drive the process. This idea was first mooted in our work on Tool Assisted Requirements Analysis [TARA paper] and further explored in the context of Viewpoints in [21]. Our work on standards compliance is in much the same vein [8]. Cass & Osterweil [2] have been looking at doing process-centred checking in the context of xlinkit using Little-JIL as the process language. We are keen to do something similar, probably using a simple reactive approach that takes advantage of distributed event monitoring.

4 Final Viewpoints: current and future challenges

We have occupied ourselves with providing a framework for separation of concerns and have made some moderately successful attacks on the associated problems of integrity that result from this. We recognise however that this is only part of the story, "having divided to conquer, we must reunite to rule" [19]. Software development requires us to be able to perform rich reasoning and analysis across the multiple views.

While tools such as xlinkit help to establish links and check rules, it is still a far cry from the reasoning and analysis that we envisage should be supported. How can we make further progress in this regard?

An example of where this challenge is evident is in the relationship between requirements and software architectures, a hot topic in current software engineering research. In the 'twin peaks' model [17, 39] we outlined our view of the co-development of requirements and architectures. Obviously, we wish to separate the expression of requirements and the representation of the software architecture. On the other hand we know that many so-called non-functional requirements drive the identification of an appropriate architecture and, vice-versa, that architectural analysis informs requirements. Handling these relationships requires more than consistency checking, important though that is. We must be able to reason about properties of the systems that we expect to build based on an integration or composition of different ViewPoint specifications. We should also be able to identify incompleteness and to perform trade-off analysis between alternative development decisions.

Given our experience and interest in software architectures, we have also investigated support for reasoning across views by exploiting the common, underlying structure of the proposed software architecture. In particular, the architecture is used as the skeletal framework upon which to hang the aspects of interest [20, 24]. In contrast to the concept of a single source of information which supports multiple views, this approach provides for the elaboration of the common underlying structure or architecture with behaviour, performance, implementation, or other information. So far this work has been loosely based on the Darwin software architecture language [22] and the LTSA (Labelled Transition Systems Analyser) [23] for behaviour analysis and animation [25], and is being extended to include synthesis from scenarios [44] and performance analysis. However, here too the relationships are limited to the granularity supported by the components and interactions inherent in the architecture rather than to any deeper notions which may be at a finer granularity or orthogonal to the architecture.

It is clear that we still lack the means to express more diverse relationships in order to reason more effectively about structured collections, or configurations, of ViewPoints. A classic example is the kind of reasoning that is needed to trade-off usability against security. The relationships between two such system requirements are complicated by the variability or lack of precision in their expression. Thus, in order to express relationships between them, individual security and usability ViewPoints need to be better understood and articulated.

Finally, returning to our example of relating requirements and architectures, we are currently exploring ways of describing the multifaceted relationships that inevitably arise between problem and solution structures in software development. We are using Jackson's Problem Frames to map out shared phenomena between requirements, problem domains, and the (software) machine to be developed [17]. Thus, our problem frames not only bridge the gaps between problems and solutions, they are also an explicit representation of some quite complex relationships between different descriptions in the software development space. We have started to explore this kind of approach to better specify and analyse security requirements and to relate them to security threats [5]. The ViewPoints framework provides a generic means of expressing the relationships between descriptions in this context. However, our experience to date indicates that identifying, expressing, and reasoning about more meaningful relationships is hard!

Acknowledgements

The ViewPoints framework was inspired by the CORE method [26] and developed in collaboration with a number of colleagues and students over many years. Instrumental in the early conceptual work, particularly during his sabbatical at Imperial College in 1989, was Michael Goedicke [10]. Steve Easterbrook has also been, and continues to be, a key developer of the framework since completing his PhD at Imperial College in 1991 [6, 7, 37, 38]. George Spanoudakis contributed to the understanding of overlaps between ViewPoints [43] and Tony Hunter to the development of techniques for reasoning in the presence of inconsistency [12, 18]. Alessandra Russo performed some of the first major case studies applying ViewPoints to NASA requirements specifications [41], and has subsequently contributed significantly to the development of tools and techniques for reasoning about evolving ViewPoint specifications [14, 42]. Wolfgang Emmerich and Christian Nentwich have been partners in the development of xlinkit [27, 28, 29]. Early work on the bridge to ViewPoints was performed by Ernst Ellmer and Andrea Zisman.

We would like to acknowledge the financial support provided by the UK SERC and then EPSRC for over a decade through a series of projects (SEED, VOILA, MISE and VOICI). The work has also benefited from financial support of the European Union in the context of projects such as ESF, PROMOTER I/II and REX.

A number of related workshops [3, 9, 45] and journal special issues [13, 15, 16] contributed to the building of an active community in viewpoints and inconsistency management research.

This paper is a retrospective on the ICSE-15 (1993) [31] paper, which received the ICSE "Most Influential Paper" Award at ICSE-2003. The original paper was subsequently revised and expanded to appear in [34].

References

1. D. Avison and G Fitzgerald, "Where Now for Development Methodologies", *Communications of the ACM*, 46(1):79 -82, January 2003.
2. A. G. Cass and L. J. Osterweil, "Requirements-based design guidance: A process-centered consistency management approach", *Technical Report 2002-024, University of Massachusetts,* Department of Computer Science, March 2002.
3. M. Chechik and S. Easterbrook, eds., *ICSE-01 Workshop on Living with Inconsistency,* Toronto, Canada, May 2001.
4. S. Clarke, W. Harrison, H. Ossher, and P. Tarr, "Subject-Oriented Design: Towards Improved Alignment of Requirements, Design and Code", *Proceedings of OOPSLA-99*, November 1999.
5. R. Crook, D. Ince, L. Lin, and B. Nuseibeh, "Security Requirements Engineering: When Anti-requirements Hit the Fan", *Proceedings of IEEE International Requirements Engineering Conference (RE'02),* Essen, Germany, 9-13 September 2002.
6. S. Easterbrook, A. Finkelstein, J. Kramer and B. Nuseibeh, "Coordinating Distributed ViewPoints: The Anatomy of a Consistency Check", *International Journal on Concurrent Engineering: Research & Applications,* Special issue on conflict management, 2(3): 209-222, CERA Institute/Technomic Publishing Co. Inc, USA, 1994.
7. S. Easterbrook and B. Nuseibeh, "Using ViewPoints for Inconsistency Management", *Software Engineering Journal,* 11(1): 31-43, BCS/IEE Press, January 1996.
8. W. Emmerich, A. Finkelstein, C. Montangero, S. Antonelli, S. Armitage, and R. Stevens, "Managing Standards Compliance" *IEEE Transactions on Software Engineering,* 25(6):836-851, Nov/Dec. 1999.
9. M. Feather, S. Fickas, and J. Kramer, eds., *ICSE-97 Workshop on Living with Inconsistency,* Boston, USA, May 1997.
10. A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein and M. Goedicke, "Viewpoints: A Framework for Multiple Perspectives in System Development," *International Journal of Software Engineering and Knowledge Engineering,* Special issue on 'Trends and Future Research Directions in SEE', World Scientific Publishing Company Ltd., 2(1): 31-57, March 1992.
11. A. Finkelstein and J. Kramer, "TARA: Tool Assisted Requirements Analysis", in *Conceptual Modelling, Databases & CASE: an integrated view of information systems development,* Loucopoulos, P. & Zicari, R. eds., 413-432, John Wiley, 1991.
12. A. Finkelstein, D. Gabbay, A. Hunter, J. Kramer and B. Nuseibeh, "Inconsistency Handling in Multi-Perspective Specifications", *Transactions on Software Engineering*, 20(8): 569-578, IEEE CS Press, August 1994.
13. A. Finkelstein and I. Sommerville, "The Viewpoints FAQ", *Software Engineering Journal,* (Special issue on viewpoints), 11(1):2-4, IEE/BCS, 1996.
14. A. Garcez. A. Russo, B. Nuseibeh, and J. Kramer, "Combining Abductive Reasoning and Inductive Learning to Evolve Requirements Specifications", *(to appear in) IEE Proceedings: Software,* 2003.
15. C. Ghezzi and B. Nuseibeh, eds, Special issue on Managing Inconsistency in Software Development, *IEEE Transactions on Software Engineering, ,* 24(11):908-1001, November 1998.
16. C. Ghezzi and B. Nuseibeh, eds, Special issue on Managing Inconsistency in Software Development, *IEEE Transactions on Software Engineering*, 25(6):782-869, November/December 1999.
17. J. Hall, M. Jackson, R. Laney, B. Nuseibeh, and L. Rapanotti, "Relating Software Requirements and Architectures using Problem Frames", *Proceedings of IEEE International Requirements Engineering Conference (RE'02)*, Essen, Germany, 9-13 September 2002.
18. A. Hunter and B. Nuseibeh, "Managing Inconsistent Specifications: Reasoning, Analysis and Action", *Transactions on Software Engineering and Methodology,* ACM Press, October 1998.
19. M. Jackson, "Some Complexities in Computer-Based Systems and Their Implications for System Development", *Proceedings of International Conference on Computer Systems and Software Engineering (CompEuro '90)*, Tel-Aviv, 8-10th May 1990, 344-351, IEEE CS Press.
20. J. Kramer and J. Magee, "Exposing the Skeleton in the Coordination Closet", *Proceedings of Coordination Languages and Models, 2nd International Conference (COORDINATION '97),* Berlin, 1997, 18-31.
21. U. Leonhardt, A. Finkelstein, J. Kramer and B. Nuseibeh, "Decentralised Process Enactment in a Multi-Perspective Development Environment", *Proceedings of 17th International Conference on Software Engineering (ICSE-17)*, Seattle, Washington, USA, 24-28th April 1995, IEEE CS Press.

22. J. Magee, N. Dulay, S. Eisenbach, and J Kramer, "Specifying Distributed Software Architectures", *Proceedings of 5th European Software Engineering Conference (ESEC'95),* 137-153, Sitges, September 1995, LNCS 989, Springer-Verlag, 1995.
23. J. Magee, and J. Kramer, *Concurrency: State Models & Java Programs,* John Wiley & Sons, March 1999, 355 pages.
24. J. Magee, J. Kramer, and D. Giannakopoulou "Behaviour Analysis of Software Architectures", *First Working IFIP Conference on Software Architecture (WICSA1),* San Antonio, Texas, 22-24 February 1999, ed. Patrick Donohoe, (Kluwer Academic Pub.), 1999, 35-50.
25. J. Magee, N. Pryce, D. Giannakopoulou, and J. Kramer, "Graphical Animation of Behaviour Models", *Proceedings of 22nd IEEE/ACM Int. Conf. on Software Engineering (ICSE-2000),* Limerick, Ireland, June 2000, 499-508.
26. G. Mullery, "CORE - a method for controlled requirements expression", *Proceedings of 4th International Conference on Software Engineering (ICSE-4)*, 126-135, IEEE CS Press, 1979.
27. C. Nentwich, W. Emmerich, and A. Finkelstein, "Static Consistency Checking for Distributed Specifications, *Proceedings of the 16th International Conference on Automated Software Engineering (ASE-01)*, 115-124, Coronado Island, CA, IEEE Computer Society Press, November 2001.
28. C. Nentwich, L. Capra, W. Emmerich, and A. Finkelstein, "xlinkit: a Consistency Checking and Smart Link Generation Service", *ACM Transactions on Internet Technology,* 2(2): 151-185, May 2002.
29. C. Nentwich, W. Emmerich, and A. Finkelstein, "Consistency Management with Repair Actions", *Proceedings of the 25th International Conference on Software Engineering (ICSE-03)*, Portland, Oregon, IEEE CS Press, May 2003.
30. B. Nuseibeh and A. Finkelstein, "ViewPoints: A Vehicle for Method and Tool Integration", *Proceedings of the 5th International Workshop on Computer-Aided Software Engineering (CASE '92*), 50-60, Montreal, Canada, 6-10th July 1992, IEEE CS Press.
31. B. Nuseibeh, J. Kramer and A. Finkelstein, "Expressing the Relationships Between Multiple Views in Requirements Specification", *Proceedings of the 15th International Conference on Software Engineering (ICSE-15),* 187-196, Baltimore, Maryland, USA, 17-21st May 1993, IEEE CS Press.
32. B. Nuseibeh, A. Finkelstein and J. Kramer, "Fine-Grain Process Modelling", *Proceedings of the 7th International Workshop on Software Specification and Design (IWSSD-7),* 42-46, Redondo Beach, California, USA, 6-7th December 1993, IEEE CS Press.
33. B. Nuseibeh, "A Multi-Perspective framework for Method Integration", *PhD Thesis,* Department of Computing, Imperial College, London, October 1994.
34. 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, IEEE CS Press, October 1994.
35. B. Nuseibeh, "To Be And Not To Be: On Managing Inconsistency in Software Development", *Proceedings of 8th International Workshop on Software Specification and Design (IWSSD-8),* 164-169, Schloss Velen, Germany, 22-23 March 1996, IEEE CS Press.
36. B. Nuseibeh, A. Finkelstein and J. Kramer, "Method Engineering for Multi-Perspective Software Development", *Information and Software Technology Journal,* 38(4): 267-274, Elsevier Science B.V., April 1996.
37. B. Nuseibeh, S. Easterbrook and A. Russo, Leveraging Inconsistency in Software Development, *IEEE Computer*, 33(4):24-29, April 2000.
38. B. Nuseibeh, S. Easterbrook and A. Russo, Making Inconsistency Respectable in Software Development, *Journal of Systems and Software,* 58(2):171-180, September 2001, Elsevier Science Publishers.
39. B. Nuseibeh, "Weaving Together Requirements and Architecture", *IEEE Computer,* 34(3):115-117, March 2001.
40. A. Rashid, A. Moreira, J. Araujo, "Modularisation and Composition of Aspectual Requirements", *Proceedings of 2nd International Conference on Aspect-Oriented Software Development,* 17-21st March 2003, Boston, US.
41. A. Russo, B. Nuseibeh, and J. Kramer, "Restructuring Requirements Specifications", *IEE Proceedings: Software,* 146(1): 44-53, February 1999.
42. A. Russo, R. Miller, B. Nuseibeh, and J. Kramer, "An Abductive Approach for Analysing Event-Based Requirements Specifications", *Proceedings of 18th International Conference on Logic Programming (ICLP-02),* Copenhagen, Denmark, 29 July-1 August 2002.
43. G. Spanoudakis, A. Finkelstein, and D. Till, "Overlaps in Requirements Engineering", *Automated Software Engineering,* 6(2):171-198, April 1999.
44. S. Uchitel, J. Kramer, and J. Magee, "Synthesis of Behaviour Models from Scenarios", *IEEE Transactions on Software Engineering,* to appear 2003.
45. L. Vidal, A. Wolf, A. Finkelstein, G. Spanoudakis, *Joint proceedings of the second international software architecture workshop (ISAW-2) and international workshop on multiple perspectives in software development (Viewpoints '96),* San Francisco, USA, ACM SIGSOFT '96, 1996, ISBN:0-89791-867-3.