Comparing Requirements Engineering Approaches for Handling Crosscutting Concerns

Ruzanna Chitchyan, Awais Rashid, Peter Sawyer

Computing Department, Lancaster University, Lancaster LA1 4WA, UK
{rouza,awais,sawyer}@comp.lancs.ac.uk

Abstract. A number of requirements engineering (RE) approaches have focused on addressing broadly scoped (non-functional) properties such as security, availability, etc. More recently, several aspect-oriented requirements engineering (AORE) approaches have been proposed to tackle both functional and non-functional requirements of a crosscutting nature. In this paper, we analyse how some well-known RE approaches address crosscutting concerns. We compare these approaches with AORE approaches in order to identify the additional contributions the latter have to offer while at the same time investigating what AORE can learn from traditional RE techniques. We use our comparison to derive a set of challenges to be addressed by AORE techniques. This paper is our position statement, rather then an attempt to precisely evaluate the discussed approaches, for which several large case studies are necessary.

Introduction

Ever since the seminal work of Bell and Taylor [1] it has been recognised that software requirements "need to be engineered". Many Requirements Engineering approaches have been developed to do just that, the majority of which have focused on functionality expected from software systems. In addition, a handful of RE approaches have focused on broadly scoped properties (so-called non-functional requirements) of software systems. These non-functional requirements have been recognised to influence (or crosscut) other requirements, and mostly to be crucial for software systems' success, yet to be difficult to pin down due to their often conceptual nature.

In this paper we discuss and compare several RE approaches, looking at how they address both functional and non-functional concerns for engineering requirements. Two groups of approaches are considered in our comparison: the first group includes PREview [2], Non-Functional Requirements Framework [3], and Problem Frames [4]; the second group includes Arcade [5] and Theme/Doc [6]. The approaches in the first group have been chosen to represent contemporary RE work which recognises presence of non-functional concerns. The second group presents examples of work on Aspect-Oriented (AO) RE. AORE work not only recognises the presence of non-functional concerns, but explicitly distinguishes crosscutting (functional and non-functional) concerns as an independent group of concerns needing additional treatment procedures. In particular, these procedures include means for identifying and modularising crosscutting concerns as well as their consequent composition with other concerns. This, in turn, requires a principled support for composition.

By comparing these two groups of RE approaches, we hope to identify the additional contributions that AORE has to offer. At the same time, we investigate what are the most valuable lessons that AORE can learn from contemporary RE techniques. Finally, we use this comparison to derive a set of key challenges pertaining to the handling of crosscutting requirements to be addressed by AORE.

This paper is our position statement, rather then an attempt to precisely evaluate the discussed approaches, for which several large case studies are necessary.

We start by providing a comparison criteria (section 2), briefly introducing the selected approaches (section 3), then compare the selected approaches using our criteria (section 4), and finally conclude the paper with discussion of our findings and challenges (section 5).

2. Comparison Criteria

In order to derive criteria for assessment of Requirements Engineering approaches, we have looked at the life cycle of requirements and analysed how they arise and progress along the software development lifecycle and the life of the software product[1]. From that we have selected such comparison criteria as to support this progression through the lifecycle.

2.1. Requirement Lifecycle

During requirements engineering, in broad terms, the properties that the software must exhibit have to be elicited. The analysis of the elicited information and the associated organisational and operational context results in the synthesis of a set of requirements. These requirements need to be, as far as is possible, *correct, complete and feasible*. Achieving these qualities typically requires negotiation and *trade-off* with and between the users and other stakeholders. The set of requirements that emerges from the analysis activity needs to be *recorded* in a specification document that communicates the requirements to the people who will use them to develop the software. The documented requirements need to be *validated* to ensure that the software that they specify will meet the needs of the people from whom the requirements were elicited [7]. As development proceeds, the requirements need to be *managed so that changes are controlled.* The requirements will change and new requirements will arise during the course of the system use, so c*oping with change* is an essential need for a requirements methodology.

Thus, RE process fundamentally addresses requirement *discovery, understanding, recording, checking, communication and management* [7]. From analysis[1] of the qualities needed for supporting these activities we have selected the assessment criteria briefly discussed below.

2.2 Comparison Criteria

Identification and handling of functional and non-functional crosscutting concerns closely relates to requirement discovery and understanding. The identification part of this

[1] Due to lack of space we do not discuss the details of this analysis in the present paper.

criterion indicates if a given RE approach supports recognition of both types of crosscutting concerns. This is crucial simply because if a crosscutting concern is not detected no further treatment can be applied to it. The handling part of the criterion stipulates that merely detecting a concern is not sufficient, a process should be provided to help in treating the given concern.

Composability is the support for combining individual requirements into coarser-grained requirements. Using the AO terminology, this support should include a well defined joinpoint model and composition semantics. The joinpoint model exposes structured points through which requirements can be composed. The composition semantics provide systematic meaning to the composition.

Composability is related to understanding, checking and management activities. It is a highly desirable property for an RE approach, allowing not only reviewing the requirements in their entirety, but also detection of potential conflicts very early on in order to either take corrective measures or appropriate decisions for the next development step. The composed requirements also become valuable sources of validation for the complete system [8].

Trade-off analysis and decision support: Conflicts are inevitable between requirements. These can arise both from crosscutting and non-crosscutting requirements (e.g., due to different needs of some requirement sources, etc.). This criterion indicates if an RE approach can facilitate their identification, analysis and resolution, thus relating to understanding, checking, management, and communication activities.

Traceability throughout lifecycle: preservation of traceability between the artefacts of the software lifecycle is a necessary quality for understandable, maintainable, and manageable recorded software. This criterion could be broken into two counterparts: (a) *traceability of requirements and change to their sources of origin* and (b) *traceability between lifecycle artefacts*.

Support for mapping: most current RE approaches recognise that not all identified requirements/concerns will progress into formal design artefacts: some will map onto decisions, trade-offs, or similar. This criterion indicates if an RE approach provides support for decisions on mapping (especially for crosscutting concerns) that facilitate efficient solution choice. It is related to recording, communication, and management activities.

Evolvability: is the ease of adapting requirements artefacts due to change in the requirements or removal/addition of a requirement. Ideally, effects due to change should be localised and easy to introduce to make it viable to keep the recorded requirements artefacts up to date with the changing requirements. This supports understandability and management.

Scalability: indicates if an RE approach is equally well suited for small and large projects. This assists with management of growing projects.

3. Overview of RE Approaches

3.1 Contemporary (non-AO) RE Approaches

We have selected the three approaches below because, to a greater or lesser degree, they explicitly recognise the importance of non-functional requirements. Because of this criterion, we do not review use-case and scenario-based approaches which (in contrast) have

established themselves as a new orthodoxy in RE. Note that none of the three approaches below explicitly classify requirements into crosscutting and non-crosscutting.

3.1.1 PREview

PREview [2, 9, 10] is a Viewpoint-Oriented Approach (VOA) [11] which, as all VOA, considers the problem-related information from different perspectives (called viewpoints) [2] arising due to different responsibilities, roles, goals, etc. of the information sources. PREview complements the standard notion of viewpoints with that of organisational concerns: generalisation of the notion of goal that includes both organisational goals and constraints restricting the system or process to be analysed.

PREview concerns are identified at the very start of the RE process and decomposed into questions, constraints, or requirements. Then viewpoints are identified and recorded using provided viewpoint templates. During the requirements analysis, questions associated with concerns must be linked to all viewpoints and answered by viewpoint sources. This is followed by interaction detection and inconsistency resolution between viewpoints. In this way PREview reveals how the concerns affect the viewpoints and requirements. The decomposed concerns are used to make decisions on concern mapping to functional modules, early architectural or other decisions, etc.

PREview suggests that functionality is often negotiable, as oppose to concerns (e.g. safety in a safety-critical system).

Thus, PREview uses concerns as drivers in requirement discovery and viewpoints for actual requirements discovery.

3.1.2 Non-Functional Requirements (NFR) Framework

Central to the NFR Framework [3] is the concept of softgoal: a goal that has neither a clear-cut definition nor precise criteria for determining whether it has been satisfied. This definition fits well for non-functional requirements (which are represented as softgoals); as such a requirement can have different meaning and satisfaction criteria for different people, and even for the same person working on different projects.

In the NFR framework softgoals represent three types of entities: overall constraints on the system (NFR softgoals), concrete design or implementation solutions for constraints (called operationalising softgoals), and rationale or explanations for decisions (called claims). Softgoals are decomposed (or refined) into offspring softgoals that relate to their parents through an IsA relationship. The offsprings also contribute to their parents either positively, or negatively. The refinement methods are patterns and guidelines for decomposition based on requirements engineers' past experience and domain knowledge[2]. The framework also provides an evaluation procedure used to determine the degree to which the initial NFR softgoal is satisfied by evaluating the decisions about offsprings' satisfaction and contributions to the parents. Offsprings can have different priorities which can be used while making decisions about trade-offs and resolving conflicts.

The conflict/support relationships between non-functional requirements (e.g., cost vs. quality/ availability vs. dependability), also called correlations, are collected and catalogued. This catalogue is used to examine the cross-impact of the softgoals during trade-off analysis and alternative solutions selection.

---

[2] One can perceive these as loosely similar in role to that of design patterns [11].

3.1.3 Problem Frames

The Problem Frames (PF) approach [4] proposes to decompose complex problems into structured sets of simpler, familiar classes of common sub-problems. (The classes of common problems should be identified from the body of problem analysis work, in a fashion similar to design patters [12]). The combined descriptions/ solutions for the sub-problems then serve as the description/solution to the original problem.

The broad characteristics of common problem classes and the interactions of the real world problem domains with the intended computer system are extracted into problem frames. Each problem frame can have a number of variations, extensions and flavours. PF suggests that once the core problem frames are known to the developers, it is easier to recognise and address the variations of these classes of problems and anticipate the difficulties as well as produce efficient solutions for them.

A problem frame is represented by a problem diagram consisting of the machine (software), one or more domains, the requirement and the shared phenomena between them. Each problem frame is supplemented with a frame concern that tells what kinds of descriptions are necessary to adequately understand the given problem and what are the logical steps for its solution.

The kinds of problems addressed by PF are "… often called functional requirements …" [4]. However, PF also recognises the importance of non-functional concerns, the most important of which for PF is the composition concern. It is necessary to combine simple problem frames into composite frames that depict realistic, complex problem analysis. Hence, PF suggests that it is desirable to recognise concerns related to each problem frame and "build up a repertoire of familiar concerns" [4]. PF even demonstrates how to realise reliability as a separate problem frame. Yet, it is only possible to accomplish by mapping the potential threats to reliability onto functionality, because only functionality can be modelled by the machine element of PF. Thus, despite the importance of non-functional concerns, they are not generally systematically treated in PF, neither is the issue of crosscutting, though some pointers to treating non-functional concerns have been suggested[3].

3.2 Aspect-Oriented Approaches

AO Software Development aims to extend traditional software development techniques. Below we present two AORE approaches and outline how they extend earlier RE work.

3.2.1 AORE with Arcade

In [5, 13] a general RE process model is developed for separating aspectual and non-aspectual requirements, as well as their composition rules. A concrete instantiation of the model, using viewpoints and an XML-based composition language, along with a supporting tool, called Arcade, is also provided[4].

In the Arcade approach, aspectual requirements are similar to PREview concerns. The PREview notion of viewpoints is also utilised for requirement elicitation. Aspectual requirements crosscut the viewpoints. Both of these are represented using an XML-based semi-structured framework. XML is also used for defining composition rules that employ

[3] It is worth noting that PF has been extended with AO concepts in [13]. However, this work is specific to the *security* concern only and it is not clear if it can be generalised to other crosscutting concerns. In the rest of this paper we discuss only the original PF work.

[4] From now on we will use 'Arcade approach' to refer to this instantiation.

informal (concern-specific) actions and operations reflecting how aspectual requirements affect (or advise) groups of viewpoint requirements that they crosscut. The set of composition rules is extensible; new problem specific rules can be created when required.

The validation of composition relationships, interaction and trade-off point detection process is assisted via the Arcade tool. Once detected, conflicts are resolved via priority fuzzy value assignment where the more important requirements receive higher priorities, thus getting preference in conflict resolution. Finally, the aspectual requirements are mapped to decisions, functions or design aspects.

The most valuable and novel contributions of this approach are its ability to separate and then compose crosscutting and non-crosscutting requirements, flexible set of composition operators, and original requirement presentation in XML. It also provides a thorough procedure for conflict identification and resolution. Arcade's support for requirements mapping, though very useful, lacks the rigor and thoroughness of PREview or NFR concern analysis. On the other hand, recent work on Arcade [8] allows the tracing and verification of requirements and trade-offs to the later stages of software lifecycle.

Thus, using the AO terminology, in Arcade the concerns serve as aspects that crosscut viewpoints; viewpoint requirements serve as joinpoints; and composition operators allow quantification over viewpoint requirements during composition.

3.2.2 Theme/Doc

Theme/Doc [6, 14] is the RE part of the Theme approach. Them/Doc supports "aspect identification and analysis in requirements documentation" where aspects manifest themselves as "descriptions of behaviours that are intertwined, and woven throughout" [6]. Thus, it is aimed at later stages of RE, when at least an initial Requirements document is available for lexical analysis.

The approach is supported by the Theme/Doc tool which is quite central to the approach because its analysis and steps are based on visualisations from the tool. The tool receives as an input the requirements document along with a list of action words selected by the requirement engineer. Action words and requirements are then depicted in the tool's action view as boxes; actions are linked with a line to the requirements that they appear in. If an action is linked to more then one requirement it could potentially reflect a crosscutting occurrence. The requirements engineer revisits the links, re-groups the words and requirements and clips the links between action words and requirements to which they provide secondary functionality. Clipped links are replaced with a decorated link. This process leads to grouping of requirements around main (base) and secondary (aspectual) functionality (called themes). The themes are then organised in accordance with order of composition: the base themes form the 1st level, themes that are linked through clipped-decorated links to the base themes only form the 2nd level, themes crosscutting base and 2nd level form the 3rd level and so on (this is the clipped action view of Theme/Doc tool).

Several other views are provided by the tool to help plan for design and mapping of the requirement views to Theme/UML – the design counterpart of the Theme approach, as well as verify the mapping between design and requirements.

Thus, some themes, that encapsulate aspectual functionality, crosscut other themes where individual requirements in the theme serve as joinpoints; the order of composition for themes is reflected by the levels of themes in clipped action view.

Currently the work on Theme/Doc approach is focused on addressing the scalability issue of the approach [15].

4. Comparison

Having outlined the RE approaches (section 3), we now investigate how well they perform against our comparison criteria (presented in section 2).

4.1 Identification and handling of both (crosscutting) functional & non-functional concerns

In the short descriptions of the contemporary RE approaches (section 3.1), we have noted that each approach has mainly focused on either functional (PF) or non-functional (NFR, PREview) concern identification and treatment. PREview and NFR recognise the crosscutting impact of non-functional concerns, but do not consider similar characteristics for functionality and neither does PF. In addition, PF does not provide a systematic support for non-functional crosscutting concerns either (see section 3.1.3).

On the other hand, the AO approaches (section 3.2) come with a promise to treat both types of concerns equally well. However, they have yet to demonstrate that this aim is realistically achievable. The Arcade approach, for instance, has a generic enough mechanism for concern handling which will suit both functional and non-functional concerns, but it is not clear how the crosscutting functional concerns should be identified. Similarly, the Theme/Doc approach has based aspect identification on using action words, but non-functional requirements often do not have any action associated with them. It is suggested that in such cases the requirements can be re-written to include action words, however this assumes that such requirements can be identified by the requirements engineer and related to other requirements. All this makes Theme/Doc less suitable for non-functional crosscutting concern identification and treatment.

Thus, the key issue of identification and handling of both functional and non-functional crosscutting and non-crosscutting concerns has not yet been adequately addressed by any of the discussed approaches.

4.2 Composability

Problem decomposition is a natural way of reducing complexity, and it is indeed the path taken by all the approaches discussed above. PREview decomposes per viewpoints and concern, NFR – per softgoal, Problem Frames – per frame, Arcade – per aspect and viewpoint, and Theme/Doc – per theme. On the other hand, the need for composing the decomposed requirements/concerns has been addressed less thoroughly.

Neither PREview nor NFR consider composition. In PF composite frames are built by joining simple frames through their common domains. Since each frame has its own description of a domain and PF does not have well defined composition semantics for them, the composition often requires solution-level information, thus departing from requirements to design and implementation issues. As a result, the composition is often an ad hoc process, and though PF has a potentially usable joinpoint model, these joinpoints are inconsistent between different frames. This approach does not allow quantification for composition either. Nevertheless, we should note also that the most recent work on PF [16] has started to look at this issue.

Theme/Doc in its clipped action view provides the order for theme composition, but it does not produce a view for composed requirements. Instead it postpones actual composition to design level, where Theme/UML composition semantics are used. Hence, while Theme/Doc does have a clear joinpoint model (with requirements in theme joinpoints) the composition semantics at requirements level are missing.

Arcade supports requirement composability through a clear joinpoint model (where requirements in a viewpoint are joinpoints) and well defined composition semantics provided through its composition rules and operators. Moreover, the composition semantics are adaptable for each problem, as the set of composition operators is extendable.

4.3 Trade-off analysis and decisions support

All the above discussed approaches recognise that concerns (as well as goals and viewpoints) can support or contradict each other. However, not all these approaches provide sufficient support for resolving such conflicts and trade-offs for alternative choices.

NFR provides good support via correlation catalogues, claims, contributions, and offspring priorities; so does Arcade through conflict detection, weight assignments, and contribution tables. The recent work on PF only provides priority assignment [16] (e.g., to events). PREview imposes a priority hierarchy in which concerns take precedence over viewpoint requirements. Theme/Doc does not provide any explicit support.

4.4 Traceability through software lifecycle

With regards to traceability of requirements and change to their sources of origin, PREview, NFR, and Arcade keep clear reference to the sources of requirements origin and change. On the other hand, PF and Theme/Doc do not attend to this issue.

With regards to traceability between lifecycle artefacts, PREview concerns end up scattered across viewpoints in the requirements elicitation stage and across requirements specification and design artefacts later on. On the other hand, Theme/Doc keeps clear links between requirements artefacts and their design incarnations due to direct match between its RE and design models as well as the major action and theme views of the tool. NFR relates its softgoals to the appropriate functional requirements via design decision links while the operationalisations relate to design decisions via operationalisation links, thus, linking requirements and designs as well as functional requirements with the non-functional ones which they are related to. PF could be considered to partially support this criterion, as, although it does not provide explicit means for traceability preservation, the approach is focused on understanding the links between the real world and the machine, which become documented in the problem diagrams and annotations. In addition, the solution patterns can be traced to the problem frames via frame concerns. Finally, Arcade records what type of architecture or design artefact a concern transforms to (e.g. decision, function, etc.), and can trace it to architecture, design, and implementation via the PROBE framework [8].

4.5 Support for mapping

Amongst our considered approaches support for mapping is provided through templates for Concern Decomposition, as in NFR and PREview methods, as well as through guidelines, as partially provided by PF and Arcade.

On the other hand, although Theme/Doc does not provide explicit mapping support, all the concerns modelled at the requirement stage are very cleanly mapped from Theme/Doc to Theme/UML due to closeness of their requirements and design models.

4.6 Evolvability

With regards to evolvability, in PREview removal or addition of a concern will result in heavy changes across all viewpoints requirements since the small number of concerns is assumed to be stable (a legacy of PREview's origin in the dependability domain). Similarly, in PF a change in requirements can result in a change as serious as unsuitability of previously selected frame decomposition.

In this respect the NFR framework is somewhat better, as it strives to first consider the non-functional requirements in isolation, before correlating them together. Thus, with NFR in case of change only the changed concern needs to be re-analysed, together with the correlations. Similarly, Arcade approach uses a number of cross-reference tables to identify impacts and contributions of concerns. In each of these tables the changed line/column will have to be reviewed, without affecting the rest of the tables. The composed requirements can also be updated by simply updating the composition specification.

Evolution in Theme/Doc requires re-generation of the views involving decision as to which theme does the new requirement belong with, and how the change/addition of a theme affects composition ordering. This process can also be supported through the traceability functions of the tool.

4.7 Scalability

Although all the approaches discussed above strive to be scalable, in all cases, (at least some aspects of) the proposed methodology does not scale adequately. In PREview, for instance, the number of concerns usable per project is limited to about 6; the Softgoal Interconnection Graph in NFR, tables in Arcade, and graphical representation of concerns in Theme/Doc all become unmanageably large, and so does the number of problem frames for relatively large problems in the PF approach.

In some cases the issue of scalability can be reduced to tool support, but it is not always sufficient, particularly when human interpretation or decision is necessary.

The results of the above discussion (representing our position on the discussed approaches) are summarised in Table 1 below. The summary is for the current state of affairs, as in many cases (particularly AO) approaches have potential for improvement.

Criteria	PREview	NFR	PF	Arcade	Theme/Doc
I & H	Non-functional crosscutting; functional non-crosscutting	Non-functional crosscutting	Functional non-crosscutting	Non-functional crosscutting; functional non-crosscutting	Functional crosscutting and non-crosscutting
Com	Not considered	Not considered	Partial support through domains	XML-based composition rules, actions and operators	Partial support through sequence of composition
Trade	Partial support through priority of concerns over viewpoint requirements	Correlation catalogues, claims, contributions, offspring priorities	Partial support through priority assignment to events and actions.	Conflict detection, weight assignment, contribution tables	Not considered
Trace	*Source traceability*: viewpoints. *Artefact traceability:* not considered.	*Source traceability*: claims. *Artefact traceability:* design decision, operationalisation links	*Source traceability*: not considered. *Artefact traceability:* partial implicit support through problem diagrams and annotations, frame concerns	*Source traceability*: viewpoints. *Artefact traceability:* PROBE framework	*Source traceability*: not considered. *Artefact traceability:* implicit support through direct match of RE and design models
Map	Partial support through templates	Partial support through templates	Partial support though guidelines	Partial support though guidelines	Implicit support thogh direct match of RE & design models
Evolov	Not considered	Partial support: through separate softgoal graphs	Not considered	Cross-reference tables, separated composition	Partial support through separation per theme
Scal	Not considered	Partial support though tools	Partial support though use of larger problems per frame	Tool support; use of text & XML; separate composition and concerns	Not considered

Table 1. Aggregated Summary of Comparisons: our position.
Legend: I&H identification and handling of concerns; Com composability; Trade trade-off analysis and decision support; Trace traceability; Map support for mapping; Evolv evolvability; Scal scalability.

5. Discussion and Challenges

The discussion above reveals that, we believe, there is still a wide range of outstanding issues (such as scalability, evolvability, etc.) for AORE to address. Due to lack of space we constrain this section to a few crucial challenges from that range:

- *Viability Challenge*: While comparing the selected RE approaches, we recognise a striking difference between the two groups in their corresponding aims. The first sets out to solve RE problems from a specific perspective: non-functional requirements, or concerns, or specific problem frames. The second intends to support both functional and non-functional crosscutting and non-crosscutting concerns. However, as we have discussed above, these approaches have not yet convincingly demonstrated that they can deal with identification and treatment of all concerns. Besides, to date there are no significant reports on application of these approaches. Thus, the first challenge for AORE is to prove that a generic AORE approach addressing both crosscutting functional and non-functional requirements is viable.
- *Composition Challenge*: It is also notable that the discussed approaches that address non-functional requirements produce elaborate structures for their decomposition (e.g. NFR), while the AO approaches do not. We believe that, in order to deeper understand these concerns and their inter-relationships and influence, the AO work should lean on these decomposition structures but complement them with equally elaborate composition and mapping support. Although some initial work for this has been done [5], it has just begun and the challenge for AORE is to develop thorough composition semantics for requirements and more elaborate joinpoint models.
- *Usability Challenge*: The discussion in sections 2 and 3 also points out that the non-AO approaches rely on patterns. For instance, problem frames are deduced from the patterns of problems, refinements in NFR are based on patterns of decomposition, etc. We believe that such patterns can be of great value in helping to disperse use of AORE, and it is a challenge to AORE to develop and utilise patterns of crosscutting requirements.
- *Multidimensionality Challenge*: Yet another outstanding issue is whether AO approaches will be able to move towards non-dominant decomposition of requirements. In all RE work to date there is a dominant decomposition structure present (e.g., concerns in PREview, softgoals in NFR). All other concerns are considered from the perspective of how they relate to the dominant ones. Thus, the challenge to AORE is to put forward the first multidimensional RE approach.

Work on addressing these (and other) challenges is already underway within the Analysis and Design Lab of AOSD-Europe Network of Excellence, as well as the MULDRE project.

Acknowledgement: This work is supported by European Commission Grant IST-2-004349: European Network of Excellence on AOSD (AOSD-Europe) and EPSRC Grant EP/C003330/1: Multidimensional Analysis of Requirements Level Trade-Offs (MULDRE).

References

[1] T. E. Bell and T. A. Taylor, "Software Requirements: Are they Really a Problem?" presented at International Conference on Software Engineering, San Francisco, USA, 1976.

[2] I. Sommerville and P. Sawyer, "PREview Viewpoints for Process and Requirements Analysis," Lancaster University, Lancaster REAIMS/WP5.1/LU060, 29 May 1996.

[3] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Funcitonal Requirements in Software Engineering*: Kluwer Academic Publishers, 2000.

[4] M. Jackson, *Problem Frames: Analyzing and Structuring Software Development Problems*: ACM Press, 2001.

[5] A. Rashid, A. Moreira, and J. Araujo, "Modularisation and Composition of Aspectual Requirements," presented at 2nd International Conference on Aspect Oriented Software Development (AOSD), Boston, USA, 2003.

[6] E. Baniassad and S. Clarke, "Theme: An Approach for Aspect-Oriented Analysis and Design," presented at International Conference on Software Engineering, 2004.

[7] P. Sawyer, "Software Requirements," in *Software Engineering*, vol. 1, R. Thayer and M. Dorfman, Eds., 3 ed: IEEE Computer Society Press, to appear.

[8] S. Katz and A. Rashid, "From Aspectual Requirements to Proof Obligations for Aspect-Oriented Systems," presented at International Conference on Requirements Engineering (RE), Kyoto, Japan, 2004.

[9] I. Sommerville and P. Sawyer, "Viewpoints: Principles, Problems and a Practical Approach to Requirements Engineering," Lancaster University, Lancaster CSEG/15/1997, 1997.

[10] P. Sawyer, I. Sommerville, and S. Viller, "PREview: tackling the real concerns of requirements engineering." Cooperative Systems Engineering Group, Computing Department, Lancaster University, Lancaster, Technical Reports No: CSEG/5/96, 1996

[11] A. Finkelstein and I. Sommerville, "The Viewpoints FAQ."

[12] E. Gamma, R. Helms, R. Johnson, and J. Vlissdes, *Design Patterns: Elements of Reusable Object-Oriented Software*: Addison-Wesley, 1995.

[13] A. Rashid, P. Sawyer, A. Moreira, and J. Araujo, "Early Aspects: a Model for Aspect-Oriented Requirements Engineering," presented at International Conference on Requirements Engineering (RE), Essen, Germany, 2002.

[14] E. Baniassad and S. Clarke, "Finding Aspects in Requirements with Theme/Doc," presented at Workshop on Early Aspects (held with AOSD 2004), Lancaster, UK, 2004.

[15] E. Baniassad and S. Clarke, "Investigating the Use of Clues for Scaling Document-Level Concern Graphs," presented at Workshop on Early Aspects (held with ECOOP 2004), Vancouver, Canada, 2004.

[16] R. Laney, L. Barroca, M. Jackson, and B. Nuseibeh, "Composing Requirements Using Problem Frames," presented at Requirements Engineering Conference (RE 2004), Kyoto, Japan, 2004.