# Multi-Perspective Requirements Engineering for Networked Business Systems: A Framework for Pattern Composition

Zlatko Zlatev, Maya Daneva, and Roel Wieringa

University of Twente, Department of Computer Science, Information System Group
P.O. Box 217, 7500 AE Enschede, the Netherlands
Z.V.Zlatev@ewi.utwente.nl
http://www.cs.utwente.nl/~zlatko
m.daneva@utwente.nl
roelw@cs.utwente.nl

**Abstract.** How business and software analysts explore, document, and negotiate requirements for enterprise systems is critical to the benefits their organizations will eventually derive. In this paper, we present a framework for analysis and redesign of networked business systems. It is based on libraries of patterns which are derived from existing Internet businesses. The framework includes three perspectives: Economic value, Business processes, and Application communication, each of which applies a goal-oriented method to compose patterns. By means of consistency relationships between perspectives, we demonstrate the usefulness of the patterns as a light-weight approach to exploration of business ideas.

## 1 Introduction

In the past decade, the opening of the Internet for commercial use has caused many revolutionary changes and brought to the business world many disruptive possibilities [2]. Businesses have experienced considerable changes in their value propositions and needed to rethink their business models to respond to the opportunities and challenges of a widely available and cheap ICT (Information and Communication Technology) infrastructure. This situation created needs for requirements engineering (RE) methods that combine exploration of new business ideas and redesign of outdated business models.

Businesses are at constant pressure to innovate and adapt to the changing environment, while at the same time reuse as much as possible of their past investments. They need to respond fast in order to keep or create competitive advantages [10]. For these reasons, businesses need RE methods that reuse existing design knowledge and produce cheap evaluation specifications in terms of time and effort [8]. Furthermore, RE methods have to assist innovation enabling design freedom and late resolution of potential conflicts in the specification [5]. And last but not least, businesses differentiate based on the quality of service they offer with their products [10], which also have to be represented in the RE methods.

The above described business needs motivate a RE framework that has the following properties:

- Reuse of existing design knowledge: the framework needs a notion of design patterns[1] as certain solutions reappear in various business models. By means of the patterns, the framework speeds up the development process and reduces its costs [11].
- Quick and effortless exploration: the framework should imply a light-weight approach to throw-away prototypes. Options' viability must be checked rapidly and cheaply [8].
- Orientation towards legacy systems: the framework needs to account for systems set up in place as most of the redesign projects build upon existing systems [10].
- Handling inconsistent specifications: the framework needs to tolerate inconsistency by postponing resolution of conflicting specifications. Management of inconsistencies, caused by stakeholders' conflicting goals and partial specifications, increases the design freedom and enables innovative solutions [5].
- Orientation towards non-functional requirements: the framework should take into account non-functional requirements as these determine the quality of service they offer [10].

Various RE approaches have been proposed [6, 8, 12, 13, 16, 17, 19, 23, 25], each of which with a different focus of analysis. Our research [24] indicates that the present RE methods support only partially and with wide variety the above five criteria.

In this paper, we propose a framework that helps business and software analysts evaluate business systems requirements. We define business systems as software applications used to support particular business processes. A business system also includes the people and organizational entities that perform certain activities to deliver value to clients. The application domain we are focused on is electronic commerce (e-commerce). Specifically, we are looking at electronic intermediaries offering negotiation services. Nevertheless, we believe our framework is applicable in the general sense of any e-business.

Further, we outline a framework which considers the five criteria listed above. Our proposal builds on libraries of three different types of design patterns. Each type of patterns predetermines a single perspective on the system. Within perspectives, patterns are composed with the help of non-functional requirements. As a result, partial views of the system are developed. They are integrated and checked for consistency.

The remainder of this paper is organized as follows: Section 2 defines the basic concepts of our framework and their relationships. Section 3 explains in detail our framework and illustrates it with an example. Section 4 reviews related work and compares it with our approach. Section 5 summarizes the framework properties and motivates its use as means to demonstrate the usefulness of business design patterns.

[1]Design patterns is used in a sense of Gamma at al.'s object-oriented design patterns [7]

## 2 Basic Concepts

This section outlines the key concepts of our framework in terms of definitions and essential properties.

**Patterns**. Patterns are fragments of designs, which occur repeatedly in particular context. These have specific properties which are persistent throughout their use in different systems. Patterns are reused in similar contexts as solutions to specific problems.

**Goals**. Goals are stakeholders' objectives. They may be functional or non-functional. Functional goals, also called hard goals, require certain capabilities to solve a particular problem. Non-functional goals, also called soft-goals, are concerned with the quality attributes of the solution.

**Pattern selection**. The selection process of a pattern as a potential solution to a problem is based on its degree of satisfaction with respect to a stakeholders' goal. Functional goals filter out patterns that are irrelevant for the particular problem and non-functional goals determine the best match in a trade-off between conflicting objectives.

**Perspectives**. Multi-perspective system development is a wide-spread approach for design of complex system. It is a well motivated choice in cases of projects that cross several domains of expertise or involve many stakeholders, or both. Perspectives, also called viewpoints [6, 14], help look at the system in a way restricted in expressiveness by: (i) stakeholders' need of information, (ii) domain knowledge, (iii) design methodology, (iv) knowledge representation technique, and (v) a set of system's properties.

**Views**. Views are partial specifications of the system developed within one perspective. A view may consist of more than one model; nevertheless, we assume for simplicity that a view contains only one model.

**Consistency**. A set of specifications is consistent if it is possible to find an example of a system that conforms to all specifications [12]. In our particular case, two views are consistent if it is possible to build at least one system that implements correctly the specified requirements.

## 3 The Framework

We propose a framework with three perspectives, including: (1) Economic value, in which we model the creation of value among networked businesses and analyze the incentives for them to take part in such a network; (2) Business processes, in which we model the coordination of activities realizing exchanges of economic values; and (3) Application communication, in which we model data exchanges among the components of the information system supporting the business.

We consider the proposed three perspectives the right balance for a light-weight framework but yet providing with the necessary analysis.

In this section, we describe our framework in detail. We begin with the components within a perspective. We specify the sequence of steps that guides the development of a concrete view. After that, we present the mechanism of view integration. We describe the established inter-view relationships and outline the approach to consistency.

### 3.1 A Framework Perspective

**Fig. 1** gives an overview of the design knowledge transformations within one perspective. Additionally, it includes preparation steps which are prerequisites for the framework. All three perspectives have the same structure; therefore, the explanation abstracts from perspective specifics such as modeled system properties or notation.

From left to right, **Fig. 1** presents the development process of one view. The column Business models and the Extract patterns arrow (far left in the figure) represent the discovery of design knowledge which is further utilized in the perspective. The Patterns column (middle in the figure) denotes design fragments ready to be composed; i.e. the input for the development of a single perspective. The right half of **Fig. 1** shows the additional knowledge needed to select and combine patterns. Further, we explain the details following the structure of the brief summary above.

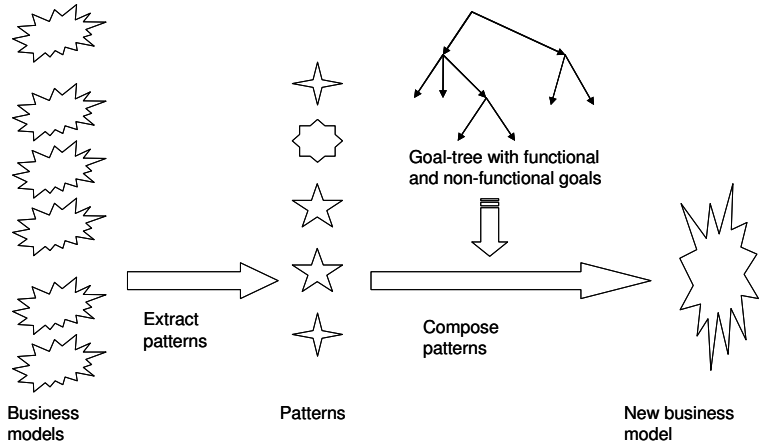


**Fig. 1.** Internal structure of perspectives

1. *Existing business models*. The star-like shapes on the left side in **Fig. 1** represent models of current businesses. These are of different types but operate in similar markets or offer similar products. Thus, we assume that if they are situated in similar environments, they would have similar problems. Therefore, we expect that their solutions have certain commonalities.
2. *Pattern Extraction*. Given models of existing business, we apply commonality analysis to identify what seem to be common patterns for the models in each view.

3. *Library of patterns*. From the collection of business models, we extract patterns (in the sense of Section 2.) We describe the patterns in a template and put them in a library. An important field in the template is the *Goal* field, as the definition of a pattern requires a pattern to be a solution to a problem. The goal is an objective and a functional specification. Additionally, we annotate our patterns with graphs in which the nodes represent non-functional requirement and the arcs show relations between non-functional requirements. We do this because we need additional information on some quality attributes of the patterns [25]. The identification, annotation and classification of patterns are preparation steps delivering ready-to-use libraries available to architects before the start of the actual development of new business models.
4. *Goal-tree*. We take the specification of functional and non-functional requirements as the input point for selection of patterns. We adopt a goal-oriented approach towards a new business. We assume that the appropriate stakeholders are able to specify a single business goal that characterizes the new business. Further, we assume that this goal can be decomposed into sub-goals. The sub-goals include hard-goals which represent functional requirements and soft-goals which represent non-functional requirements. The goal decomposition forms a tree with different relationships between leaves and branches, e.g.: a goal supports another goal, a goal prevents another goal, a goal weakens another goal, or a goal is an alternative of another goal. We take advantage of these relationships later in the composition phase.
5. *Pattern Composition*. Our starting point is: (1) a library of patterns and (2) a goal-tree of a new business idea. Each pattern in the library is annotated with a graph expressing the degree of influence between functional and non-functional properties of the pattern. The goal-tree, on the other hand, has similar relationships between hard and soft goals. Following Gross and Yu's approach [9], we select the building block for the new business model from the library of patterns. The selected bunch of design fragments needs additional work to become a coherent design. We employ some heuristics to connect the fragments and achieve simple consistency [25]. Despite that, we do not require the composed patterns to form an internally consistent and completely developed view. We intentionally leave design freedom in order to accommodate changes due to integration with other viewpoints.

### 3.2 Example of Pattern Composition

We illustrate the development process within the value perspective by means of an example based on our previous work [25]. Let us assume that we need to build a new networked business model with a provider and a consumer of a particular service. An example goal-tree of the provider is shown in **Fig. 2**. We focus only on one of its goals, namely Allow reservations. It is a functional goal which is sub-divided into two non-functional goals. One of these, Late Payment, requires possibility for consumers to pay after service consumption, whereas the second, Secure resources, requires payment in advance to avoid losses from unconsumed services. Obviously, the goals are conflicting and the best trade-off will select the best pattern.

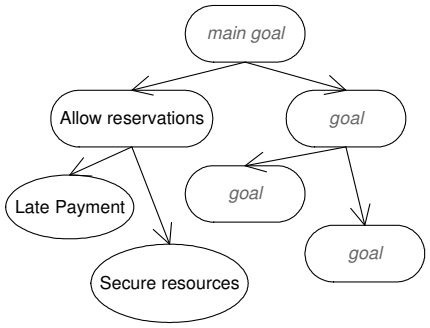


**Fig. 2.** Service provider goal-tree, where: arrows mean goal decomposition; ovals mean functional goals; and ellipses denote non-functional goals

**Fig. 3**, (a) and (b), shows two alternative value exchange patterns that both satisfy the functional goal Allow reservations. They differ in their non-functional properties. The pattern in **Fig. 3** (a) shows a transaction where a reservation is made only in case money is given in advance of service consumption. Whereas, the pattern in **Fig. 3** (b) includes a bank which plays a role of a trusted third party. The consumer uses the Credit letter issued by the bank to guarantee its future payment for the service consumption. Importantly, the consumer does not pay in advance.

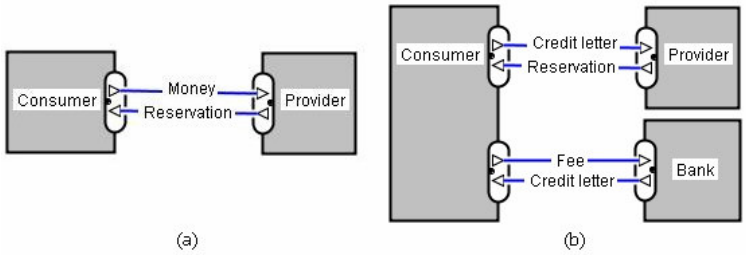


**Fig. 3.** Reservation patterns, using the value modeling notation *e³-value* [8]

The analysis shows that the pattern from **Fig. 3** (b) matches better our non-functional goals from **Fig. 2**. It allows late payment by the consumer and at the same time secures provider's resources. In contrast, the pattern from **Fig. 3** (a) satisfies only the Secure resources goal. Therefore, we select the pattern from **Fig. 3** (b) and include it in the new business model. A value model usually includes more value patterns but for illustration purposes one would suffice.

### 3.3 Perspectives Integration

Each perspective in our framework is organized in a way to allow integration with other perspectives. Each deliverable of a perspective analysis is a model (also called a view) that is potentially underspecified and therefore is likely to get changed in order

to align to requirements from other views. Below, we describe how the three views are integrated into a coherent framework.

**Integration**. **Fig. 4** gives an outlook of our framework, which looks like three rays directed to a common focal point. Each ray represents a perspective which produces a view shown in the middle of the diagram. The figure shows our concrete perspectives; we model networked business systems from *Economic value*, *Business processes* and *Application communication* perspectives. The three perspectives are integrated through the three views they deliver. The relation between them is one of consistency. As long as they are specifications of one system, they must contain no contradicting requirements.

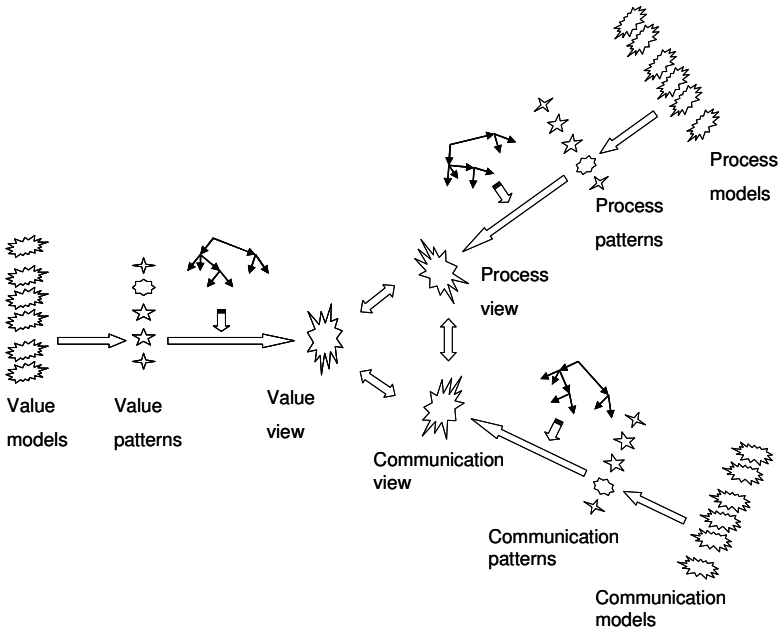


**Fig. 4.** Integration of Economic value, Business processes and Application communication perspectives

**Consistency**. Establishing consistency between views is the major step for integrating views. At this stage of development, consistency comes in two types: consistency between pairs of views and global consistency among all views. The decentralized development of each perspective prevents us from establishing global consistency. (We come to that point in the next sub-section, where we discuss goal-trees for different perspectives.) The independent development of views allows us only to check for consistency between pairs of views.

We use the consistency definition given in Section 2, which is based on Open Distributed Processing - Reference Model (ODP-RM) [12]. ODP-RM requires that there exists an implementation which conforms to all views. We implement the conformance checking procedure by means of scenarios, i.e. two views are consistent

if the executions of all their scenarios have the same effects in the system environment.

**Goal-trees**. A goal decomposition tree is a vital element in the composition of a view. The fact that we have three independent perspectives presupposes that in each of them we use a different goal-tree. We acknowledge and tolerate variations in the trees as they represent objectives of various stakeholders. Nevertheless, we require that the three goal-trees have a common root, as they represent the mission of a single business. A single goal, decomposed from various aspects in various perspectives, can be seen as a means for achieving global consistency between views.

### 3.4 Example of Consistency Between Views

In this section, we illustrate how we integrate the Economic value and Business processes perspectives by using the example case from Section 3.2. We use the model in **Fig. 3** (b) to show the value view and the model in **Fig. 5** to show the process view.

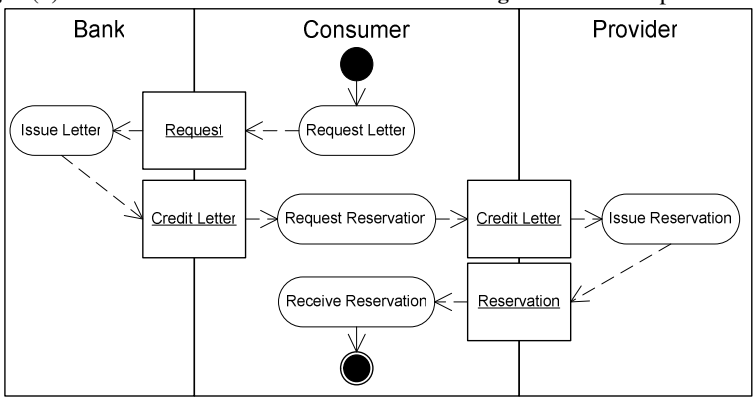


**Fig. 5.** Business process view, using a UML Activity diagram [15]

Our consistency check procedure implies two aspects. First, it rests on mapping of concept from the meta-models of the two modeling notations. Establishing this semantic relationship makes it possible to check static compatibility of the views; e.g., it gives answers to questions like whether participant the same in number and name, and are the exchanged objects the same.

Second is the dynamic aspect. We need to check if the effects of the two models are equivalent. Our consistency check procedure will be scenario based, which implies generation of all possible execution scenarios for each view. Then, for every result in the environment caused by a value scenario, there should exist a process scenario with equivalent effect, and vice versa. This means that all value objects from the value model must have been exchanged after the termination of the process model.

Even without an implemented consistency check algorithm, we can see in our example that the two views are not consistent, because the fee for the credit letter (**Fig. 3** (b)) is not present in the process model (**Fig. 5**.)

## 4 Related Work

In this section, we review related work in three different research directions which match important elements of our framework. We summarize (i) existing knowledge about business patterns and their composition, (ii) multi-perspective requirements engineering and (iii) consistency in system specification.

### 4.1 Business Patterns and Their Composition

IBM Patterns for e-business [1, 11] initiative outlines a framework, consisting at the top level of four business patterns and two integration pattern. These patterns define the high-level structure of a business. They can be further instantiated by application patterns. This approach comes very close to our communication perspective as it expresses a static view on system components including constructs like: participant, functional component, data storage and links.

Weill and Vitale [21] take another approach to business models. In their work, they identify a number of atomic business models. These are not specified explicitly as patterns; nevertheless, we asses them as such. A distinguishing property of the atomic business models is that they explicitly specify the value proposition between participants. The modeling technique includes concepts like: money flow, product flow, information flow, participant's role, and relationships. Thus, we classify the atomic models as similar to our economic value patterns.

With respect to composition, Gross and Yu [9] propose an approach for assigning qualitative scores to patterns based on their contribution to goals. These goals include soft-goals (non-functional requirements) which determine the level of satisfaction provided by patterns. Weiss [22] puts the same idea in a framework that links goal-driven and pattern-driven approaches to system design.

### 4.2 Multi-Perspective Requirement Engineering

There is a significant amount of literature on multi-perspective analysis and design of complex information systems. In our current work [24], we study and evaluate various approaches. Below, we provide a short summary of two of them:

**ArchiMate** [19]. ArchiMate is an integration framework for architecture models from different domains. It offers a common specification language for all its layers and aspects, which allows for mapping between the concepts in 15 views. The consistency is achieved by two-way translation from every notation to the ArchiMate language; every modeling notation is wrapped up with a mapping layer. Despite the common language, we consider the number of viewpoints and the overlaps between them as a drawback. Furthermore, the framework does not allow inconsistency.

**Reference Model for Open Distributed Processing (RM-ODP)** [12]. RM-ODP is a framework for architecture specification of large distributed systems. The specification of a system in terms of RM-ODP has five separate but interrelated viewpoints: Enterprise, Information, Computational, Engineering and Technology

viewpoints. With respect to consistency, the framework requires translation of models from one representation to another but does not specify inconsistency resolution. RM-ODP offers a distributed bottom-up approach with relatively limited number of viewpoints. The drawback is that some of the views are related with refinement-abstraction relationship, which violates their independence.

### 4.3 Consistency

There are several approaches to inter-viewpoint consistency checks. Here, we list three, namely a consistency check with direct translation, with canonic representation and with meta-representation.

The direct translation consistency check [3] is based on translations between views. In case of only two views, one of them is translated into the modeling notation of the other. Further, the consistency is checked within the second viewpoint. In case of n views, n*(n-1) translations and checks are needed.

The canonical representation approach [4] selects one modeling technique (not necessarily one used in any perspective) in which terms all consistency checks are made. This approach requires less translation between view (in case of n views 2*n translations). This advantage is a trade-off with respect to the expressiveness of the canonical representation technique.

The meta-representation approach [18, 20] does not require translations between modeling techniques. It specifies relations between meta-modeling and modeling concepts from each modeling technique. These relations must hold between the modeling concepts and their instances in each view.

## 5 Discussion and Conclusion

In a nutshell, our framework can be described as a multi-perspective pattern-driven framework for RE of networked business systems. We highlight its key characterizing aspects:

- Multi-perspective-ness. Our framework suggests developers to consider three perspectives: Economic value, Business processes and Application communication. This is (i) to achieve views on the system from viewpoints with little overlap and (ii) to reach a trade-off between a detailed system specification methodology and a light-weight throw-away-prototype analysis framework. Moreover, only the subject domain of the perspectives is fixed; the modeling technique, the methodology and the knowledge source are left unspecified. This accommodates flexibility when specific issues or expertise are at hand. Finally, adding or removing perspectives is not restricted; the overhead in adding a new perspective is the definition of consistency relationships with the other perspectives.
- Pattern-driven-ness. The core of our framework is a library of patterns. The patterns are extracted from real-life networked businesses and catalogued according to functional and non-functional goals they satisfy. In such a way,

a new design can be rapidly composed from fragments tested in existing systems. Moreover, the shortcut link to stakeholders' needs (through patterns) allows to gain a quick overview of the system under development.

- Requirements engineering orientation. Our framework crosses two domains, the business and the information systems domain. We see it as a framework for analysis of business problems and exploration of ICT solutions.
- Networked businesses orientation. Based on the pre-defined perspectives, we aim at the analysis and (re)design of businesses that use extensively ICT.

Our framework does not suggest a particular goal-oriented technique for pattern composition. Any method that selects and puts together patterns, and resolves conflicts between them will be a good choice for the architect. Nevertheless, for the purpose of our research, we selected the i* goal-oriented RE method as a suitable technique [23]. It allows us to manipulate patterns by distinguishing them based on functional and quality attributes.

Our research lies in the domains of business and ICT. We are interested in the reuse of design knowledge encapsulated in the value, process and communication patterns. In this broader research, our framework is used to demonstrate the usefulness of patterns as means for exploration of new business ideas. The framework is our vehicle to show how patterns are composed into views, and how views reinforce themselves with consistency relationships.

Our future work includes two aspects of the framework which need elaboration. First, the selection procedure of a favored pattern in the composition step needs to be made explicit and possibly quantitative. This is a challenging task as the selection is determined by non-functional goals which are difficult to measure. The second aspect is the specification of consistency relationships. The consistency check we outlined here needs validation in several cases studies.

A particular property of the framework, namely its orientation towards legacy systems, needs additional research. Models of legacy systems can be integrated into our framework directly as fully developed views. Nevertheless, a potential preceding decomposition or identification of patterns may be beneficial.

## References


1. Adams, J., Koushik, S., Vasudeva, G. and Galambos, G.: Patterns for e-business: A Strategy for Reuse. IBM Press, October 2001, ISBN: 1931182027
2. Bower, J. L. and Christensen, C. M.: Disruptive technologies: catching the wave. Harvard Business Review, 73, 1 (January-February), 1995, pp. 43—53
3. Bowman, H., Steen, M.W.A., Boiten, E.A. and Derrick, J.: A formal framework for viewpoint consistency. Formal Methods in System Design, 21, September 2002, pp. 111—166
4. Braatz, B., Klein, M. and Schröter, G.: Semantical Integration of Object-Oriented Viewpoint Specification Techniques. In: Integration of Software Specification Techniques for Applications in Engineering. Lecture Notes in Computer Science, Springer, 2004
5. Easterbrook, S. and Nuseibeh, B.: Using ViewPoints for inconsistency management. Software Engineering Journal, Vol. 11, Issue 2, 1996, pp. 132—132
6. Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L. and Goedicke, M.: Viewpoints: A framework for integrating multiple perspectives in system development. International

Journal on Software Engineering and Knowledge Engineering, Special issue on Trends and Research Directions in Software Engineering Environments, Vol. 2, No. 1, 1992, pp. 31—58
7. Gamma, E., Helm, R. Johnson R. and Vlissides, J.: Design Patterns. Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995, ISBN 0-201-63361-2
8. Gordijn, J., & Akkermans, J.M.: Value based requirements engineering: exploring innovative e-commerce idea. Requirements Engineering Journal, Springer Verlag, 8, 2, 2003, pp. 114—134
9. Gross, D. and Yu, E.: From Non-Functional Requirements to Design through Patterns. Sixth International Workshop on Requirements Engineering: Foundation for Software Quality, June 5-6 2000, Stockholm, Sweden
10. Hammer, M. and Stanton, S.: How Process Enterprises Really Work. Harvard Business Review, Nov-Dec, 1999, pp. 108—118
11. IBM Patterns for e-business, http://www-106.ibm.com/developerworks/patterns/, accessed December 2004
12. ISO/IEC 10746-1 | ITU-T Recommendation X.901.: Open Distributed Processing - Reference Model. OMG, 1995-98.
13. Leite, J.C.S.P. and Freeman, P.A.: Requirements validation through viewpoint resolution. IEEE transactions on Software Engineering, 17, 12, 1991. pp. 1253—1269
14. Nuseibeh, B., Kramer, J. and Finkelstein, A.: A Framework for Expressing the Relationships between Multiple Views in Requirements Specifications. IEEE Transactions on Software Engineering, 20, 10, 1994
15. OMG: OMG Unified Modeling Language Specification. http://www.omg.org/cgi-bin/doc?formal/03-03-01, 2003
16. Reijswoud, V.E. and Dietz, J.L.G.: DEMO Modeling Handbook, www.demo.tudelft.nl, 1999, (http://www.demo.nl/documents/handbook.pdf)
17. Sommerville, I, and Sawyer, P.: Viewpoints: principles, problems and a practical approach to requirements engineering. Annals of Software Engineering, Vol. 3, 1997, pp. 101—130, http://citeseer.csail.mit.edu/article/sommerville97viewpoints.html
18. Sunetnanta, T. and Finkelstein, A.: Automated Consistency Checking for Multiperspective Software Specifications. Workshop on Advanced Separation of Concerns, The 23rd International Conference on Software Engineering (ICSE2001), Toronto, Ontario, Canada, 2001, May 12-19
19. Telematica Institute, ArchiMate, http://www.telin.nl/NetworkedBusiness/Archimate/ART/index.html, accessed March 2005
20. Thanitsukkarn, T.: Multiperspective Development Environment for Configurable Distributed Applications. Ph.D. Thesis, Department of Computing, Imperial College, February 1999.
21. Weill, P. and Vitale, M.: Place to Space: Migrating to Ebusiness Models. Harvard Business School Press, 2001
22. Weiss, M.: Pattern-Driven Design of Agent Systems: Approach and Case Study. Conference on Advanced Information Systems Engineering (CAiSE), LNCS 2681, Springer, 2003
23. Yu, E.: Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. Proceedings of the 3rd IEEE Int. Symp. on Requirements Engineering (RE'97) Jan. 6-8, 1997, Washington D.C., USA. pp. 226—235
24. Zlatev, Z., Daneva, M. and Wieringa, R.: A Survey of Multi-Perspective Requirements Engineering Approaches. Work in progress - internal report, Department of Computer Science, University of Twente, The Netherlands, March 2005
25. Zlatev, Z., Eck, P. van, Wieringa, R. and Gordijn, J.: Goal-Oriented RE for e-services. In: International Workshop on Service-oriented Requirements Engineering--SORE'04, Kyoto, Japan, September 2004