

Modelling and Traceability of Composition Semantics in Requirements

Ruzanna Chitchyan, Safoora Shakil Khan, Awais Rashid
Computing Department, Lancaster University, Lancaster
{rouza;shakilkh;marash}@comp.lancas.ac.uk

Abstract

Requirements composition is one of the cornerstones of Aspect-Oriented Requirements Engineering [1]. In this paper we discuss how such compositions can be represented in our Requirements Description Language (RDL). The RDL draws a number of concepts from natural language syntax and semantics; its composition semantics are entirely based on semantics of requirements interdependencies¹. We also present a visual notation for illustrating such compositions. While our RDL is dedicated to preservation of semantics of the requirements and their interdependencies, the visualisation further facilitates linking the composition semantics with their contributor requirements. The visual notation also serves as a simple means for validation of compositions with the stakeholders as well as for recording traceability of requirements participating in a composition to their sources.

1. Introduction

In surveying the contemporary works on Requirements Engineering [1], we have identified *composition* as one of the primary contributions of Aspect-Oriented Requirements Engineering (AORE). We have also pointed out that though several AORE [2, 3] and other [4] approaches provide some support for composition, more research is needed on the semantics of composition, the mechanisms for traceably capturing such semantics, and validating them with the stakeholders [1].

¹ *In a related paper [5] (also submitted to this workshop) we discuss in detail how a relatively complete set of requirements interdependencies is derived from the semantics of natural language words used in requirements themselves. We also discuss tracing these semantics into the composition-based requirements analysis.*

Yet composition semantics are inherent to the natural language description of requirements¹, as these semantics reflect the influences of some requirements on the others [5]. For instance, security requirements could be expected to broadly influence other user requirements; response time requirements may conflict with security requirements, etc. Thus, it is essential to capture such mutual influences of requirements in order to be able to carry out an informed and adequate requirements analysis. If such influences are not identified and addressed in a timely fashion, the resultant requirements specification is likely to contain contradictions and incompatibilities (i.e. conflicts).

In this paper we focus in particular on representation of requirements and their interdependencies – a necessary step before any (non-trivial) requirements analysis can commence². Our representation is mainly centred on an XML-based Requirements Description Language (RDL). The RDL draws a number of concepts from natural language syntax and semantics; its composition semantics are entirely based on semantics of requirements interdependencies [5]¹. The RDL is complemented by a simple yet intuitive visual representation to enable the requirements analyst to take the compositions back to the stakeholders for validation, hence providing validation and traceability link to the original information sources.

The main concepts of our RDL and its supported composition are outlined in section 2 of this paper. Section 3 presents the visualisation of the composition, and section 4 discusses the traceability support delivered by our visualisation approach. In section 5 we discuss the related work, and finally draw the conclusions and point to the future directions of this work in section 6.

²*We leave discussion on conflict resolution to a future paper. The issue of dependency identification is presented in detail in [5].*

2. Composition Semantics in Requirements

Initial (natural language) requirements artefacts already contain the semantics which cause their interdependencies and interaction. These are the very semantics that we need to reveal and use in composition, as **composition, in essence, is nothing else than an explicit revelation of such requirement interdependencies and interactions.**

Our approach to dependency identification is based on principles of work in Natural Language Processing (NLP). In particular, we use Dixon's [6] verb classification to ground our own verb classification and to derive a set of possible requirement interdependency types¹ [5]. Presently we use six main verb categories (Affect, Move, Rest, Communicate, Mental Action, and General Action), each of which has a number of sub-categories¹. Our RDL, outlined in sub-section 2.1 below, also uses the concepts based on natural language syntax and semantics.

In order to illustrate the concepts of the RDL we use an extract of requirements from a requirements statement document for an online auction system [7]:

“All potential users of the system must first enrol with the system.... The seller has the right to cancel the auction as long as the auction's start date has not already passed.... An auction is activated when the auction start date provided by the seller arrives...”

2.1. RDL

The most basic concepts of the RDL are that of *concern*, *requirement*, *subject*, *object*, *relationship*, *composition*, and *sequencing*. (The *composition* element and its sub-elements are discussed in the following sub-section.)

Concern is a module for encapsulating requirements. A concern can be simple (containing only requirements), or composite (containing other concerns as well as requirements). Depending on the requirements structuring approach selected, concern can be instantiated to represent the requirement encapsulation module of that approach (e.g. viewpoint, theme, use case, etc.). Each concern is identified by its name.

From our example requirements extract we can identify three concerns: Enrol, Seller, and Auction Management, each of which has only one requirement. These are depicted in Figure 1.

Requirements are defined in line with accepted Requirements Engineering terminology. They are descriptions of what services the stakeholders expect of the system, the system behaviour, and constraints and standards that it should meet [8].

Each of the sentences provided in the example extract are requirement sentences (Figure 1). Of course, it is not always necessary for each sentence to be a new requirement by itself: some may be providing details of a more general requirement, etc. This is why we have adopted a nested structure for requirements – each requirement can have any number of sub-requirements, as has initially been presented in [2].

Similar to the subject in natural language grammar, a *subject* in our RDL is the entity that undertakes actions described within the requirement sentence.

In the example extract from the auction system requirements the following subjects are identified (Figure 1): users, seller, auction, start date. These are subjects since they *undertake the action* of a requirement: users by enrolling, sellers by canceling the auction, auction by activating, and start date by passing. It is worth noting that though we have only three requirement sentences, we have identified four subjects. This is because the last sentence has a complement clause (‘when the start date provided by the user arrives’) which in turn has its own subject – start date.

```
<Concern name="Enrol">
  <Requirement id="1"> All potential
    <Subject>users</Subject> of the system must
    <Sequencing type="temporal" semantics="prior">first</Sequencing>
    <Relationship type="Move" semantics="Group">enrol</Relationship> with the
    <Object>system</Object>.
  </Requirement>
</Concern>

<Concern name="Seller">
  <Requirement id="1"> The
    <Subject>seller</Subject> has the right to
    <Relationship type="Move" semantics="Set_in_Motion">
    cancel</Relationship> the
    <Object>auction</Object>
    <Sequencing type="conditional" semantics="if">as long as </Sequencing>the
    auction's
    <Subject>start date</Subject> has not been
    <Relationship type="Move" semantics="Set_in_Motion">
    passed</Relationship>, i.e., the
    <Subject>auction</Subject> has not
    <Sequencing type="temporal" semantics="prior">already</Sequencing>
    <Relationship type="Move" semantics="Set_in_Motion">
    started</Relationship>
  </Requirement>
</Concern>

<Concern name="Auction Management">
  <Requirement id="1"> An
    <Object>Auction</Object> is
    <Relationship type="Move" semantics="Set_in_Motion">activated
    </Relationship>
    <Sequencing type="temporal" semantics="following"> when</Sequencing>
    the auction
    <Subject>start date</Subject> provided by the seller
    <Relationship type="Move" semantics="Set_in_Motion">arrives
    </Relationship>.
  </Requirement>
</Concern>
```

Figure 1: Demonstration of RDL concepts: concern, requirement, subject, object, relationship.

An *object* in our RDL is also in line with the natural language grammar object in that it is the entity which is being affected by the actions undertaken by the

subject of the requirement sentence, or in respect with which the actions are undertaken.

The objects in our example extract are (Figure 1): system (as users need to register *with* the system), and auction (as it can be cancelled). Again, it is worth noting that there are only two objects for the three requirement sentences. Thus, objects can often be omitted or implied either due to the common use of a verb/expression, or due to context, etc.

Relationship depicts the action performed by the subject on or in respect with its object(s). This does not need to be any specific type of action, but any action denoted by a verb or verb expression.

In the given examples the relationships are: enrol, since this is what the user must do with respect to the system; cancel, as this is what the seller may do in respect with the auction; passed, as the start date may pass; activate, as this is what is done to the auction; and, finally, arrive, as this is what the start date may do.

As mentioned above, we also use our own classification system to group the verbs into semantic categories each of which reflects a certain general meaning¹. The (simplified) RDL representation of the example requirements along with their verb classifications is presented in Figure 1, where *type* attribute reflects the top-level category of the verb, and *semantics* category reflects its more precise sub-category.

Sequencing reflects the ordering of stakeholder/system requirements. This element is applicable, for instance, when the stakeholders want to stipulate some temporal, conditional, or unconditional ordering of certain requirements.

In our example (Figure 1) the words *first*, *as long as*, *already*, and *when* provide examples of the temporal and conditional dependencies. The attribute structure of *sequencing* is similar to the *relationship* element, in that this element too has the *type* and *semantics* attributes which reflect the general type (temporal, conditional, unconditional) of the given word, and its more precise semantics (e.g., *prior*, *following*, *if*).

2.2. Composition²

Composition is the process of integrating requirements into a coherent set by exposing their interdependencies and ordering. Having revealed such interdependencies we can detect conflicts and

² Several other features of composition, e.g. the *outcome* element and its attributes are not discussed in this paper.

inconsistencies between requirements and take an early corrective action.

The Composition element of the RDL consists of two main sub-elements: *constraint* and *base* (c.f. Figure 3). The *constraint* element selects a (set of) requirements or concerns and defines how this (set of) requirements influences a second (set of) requirements or concerns. This second (set of) requirements/concerns is selected by the *base* element. *Base* also defines the order and conditions of the *constraint* elements' influence (or application) onto the base.

Composition utilises the above presented elements of the RDL by both directly referencing the elements (e.g., referring by name to concerns, subjects, etc., referring by type or semantics to relationships or sequencing elements, etc.) as well as by using a set of operators derived from the verb categories¹ and from the sequencing element types. In the present paper we will not be discussing operator derivation in detail, but some examples of such operators are presented in Figure 2.

Operators from Relationship Element	
Type: Move Operator: move	Takes a common role of a (thing) Moving and (optionally) of Locus – the place with respect of which the motion takes place. A <i>Causer</i> role may also be introduced in some cases.
Subtype: Set in Motion Operator: begin/end	Describes causing something to be in (a wanted or unwanted) motion (possibly concurrently with another moving object).
Subtype: Transfer Possession Operator: provide	Causing something to be in motion with respect to a Locus. Here the locus should be specified.
Subtype: Commute Operator: deliver	Refers to (the mode of) motion with respect to a definite Locus, an additional moving object with respect to which the motion occurs can also be specified.
Subtype: Group Operator: affiliate	Refers to the motion towards to a definite Locus.

(a)

Operators from Sequencing Element	
Type: Between	Indicates presence of temporal relationship between requirements. The more precise details of the relationship (e.g. whether it is concurrent or sequential, etc.) are not known.
Subtype: Prior	Indicates that requirement X is satisfied either <i>before</i> or <i>meets</i> requirement Y.
Subtype: X before Y	There is a temporal interval when the requirement X is completed but requirement Y has not started yet.
Subtype: X meets Y	There is no temporal interval between requirement X ending and requirement Y starting.
Subtype: X overlaps Y	Requirement X has commenced before requirement Y; Y commences while X is in process; X completes while Y is in process.

(b)

Figure 2: Example of Operators derived from (a) Relationship and (b) Composition elements.

The operators are used in the *constraint* and *base* sub-elements of the *composition* element. The operators derived from the Relationship categories are used to specify the how of the *constraint* element. The operators derived from the Sequencing are used to define the order/condition in the *base* element.

In Figure 3 we demonstrate the composition specification in our RDL. We illustrate two compositions:

- the first (named EnrolComposition) states that “a concern named Enrol must be affiliated before all other concerns”;
- the second (named CancelAuctionComposition) states that “a seller may cancel (end) an auction only if an auction has not begun”.

```
<Composition name="Enrol Composition">
  <Constraint operator="affiliate">concern="Enrol"</Constraint>
  <Base operator="before">all concerns</Base>
</Composition>

<Composition name="CancelAuctionComposition">
  <Constraint operator="begin/end">subject="seller" and
  relationship="end" and object="auction"</Constraint>
  <Base operator="ifNot">subject="auction" and
  [relationship="begin"]</Base>
</Composition>
```

Figure 3: Example of composition in RDL

Hence, the first example in Figure 3 shows that there is an *affiliate* relationship between Enrol concern and all other concerns of the system. The second example shows that there is a conditional *begin/end* relationship between any requirement sentence with subject *seller* that has relationship *end* to an object *auction* and any other requirement sentence with subject *action* and relationship *begin* - the *constraint* element is only applicable when the bracketed part of the base element (Figure 3) is not satisfied.

Thus, we have used the semantics of the requirements themselves, in particular their verbs and condition/order specifications in making requirement interdependencies explicit in the composition. The most powerful feature of our RDL is its use of the semantics of the natural language. These are used not only in operator derivation, but also in referencing the subjects, objects and relationships: thus, when we refer to subject of *seller* kind, we imply semantic, and not simply syntactic matching. For instance with subject="seller" all cases where subject is a seller, salesman or any other synonym of *seller* will be matched. Synonym matching can be supported either by general synonym dictionaries, or by building case-specific dictionaries for each system.

3. Visual Representation of Composition

As discussed in the section above, the RDL annotates requirements with an XML-based language. While it is expressive and powerful and serves as an effective communication medium between requirements analysts, architects and designers, the

RDL may not be easy to understand for stakeholders who are not technically-minded. In particular, the compositions in the RDL may be confusing for the typical end-users. To overcome this “understandability barrier” we have developed a simple visual representation of requirements and their compositions (i.e. mutual influences). These visual representations can then be presented to the stakeholders for validation.

Within AORE some previous research has been devoted to visual representation of requirements that facilitates modelling, assuring correct understanding of the requirements before design or implementation. For instance, the Unified Modelling Language (UML) has been widely furnished with stereotypes and metamodel extensions [9, 10]. However, such approaches lack generality as they are restricted to the models with specific extensions/stereotypes. On the other hand, approaches such as the Theme/Doc [3] defer composition to design. Thus, there is a gap in visual representation of composition in a generic (model-independent) and scalable fashion at the requirements level.

Our visual representation of composition (initially inspired by Theme/Doc [3]) aims to provide the first step towards filling this gap. It proposes pictorial notations to support the rich composition semantics of our generic RDL (described in section 2 of the paper) and facilitates compact representation of composition in a traceable way.

3.1 Visual Representation for RDL composition

Our visual representation approach aims to give a pictorial representation of the RDL composition. Thus, it must provide corresponding notations for the RDL composition operators (a sample of which is depicted in Figure 2) and the complete composition specifications, as shown in Figure 3. In accordance with the material presented about the RDL, in this section we discuss the visual notations for compositions with constraint and base operators³.

Figure 4 presents the visual notation counterparts for the temporal *base* operator *between*, the RDL description of which was presented in Figure 2(b). *Between* operator is bi-directional (i.e. it can be presented from the perspective of each of the two participating requirements separately) and it is further classified into *before*, *after*, *meets*, and *overlap* sub-operators. The visualization of *between* sub-operators

³ Discussion of outcome operators is not included in this paper.

are represented with triangular shapes which intuitively depict the sequencing of time similar to the less-than “<” and greater-than “>” comparison operators. If the time of event A is less than (<) time of B, then A would have taken place *before* B.

Between	<	Before
	>	After
	◊	Overlap
	⊗	Meet

Figure 4: Temporal *between operator* and its further classifications

In our visual notation the constraint operators are visually represented by enclosing them in double curly bracket $\{\{constraint\ operator\}\}$ (e.g., $\{\{move\}\}$, etc.). All constraint operators depicted in Figure 2(a) would appear in this way on the composition visualization.

The RDL compositions are represented in a double rectangular shaped box. Each composition box has a name that corresponds to the name of the composition specification given in the RDL. Each box also encloses constraint and base operators which establish links between concerns. The concerns, containing subjects (denoted by **S:**), relationships (denoted by **R:**), and objects (denoted by **O:**), are represented as rectangles with their RDL-given names in the top left corner (e.g., Figure 6). In cases where the roles of subject, object, or relationship are not significant for a given composition, they can be omitted from the diagram (e.g., Figure 5). The last element within a concern rectangle is the list of sources (e.g., stakeholders, user manuals, standards, etc.), that relate to the given concern, enclosed into square brackets.

Figure 5 shows the composition between ‘enrol’ concern and all of the other auction system concerns (composition specified in the RDL in Figure 3). The ‘Enrol’ concern originates from the owner of the auction system (referred to as Auctioneer), and all other concerns are restricted by this concern which is specified by ‘[*]’. The composition for EnrolComposition is visually represented by $\{\{affiliating\}\}$ the enrol concern before (as pictorially shown by ‘before’ sub-operator of *between operator*) all auction system concerns (pictorially shown by concern having name ‘all concern’).

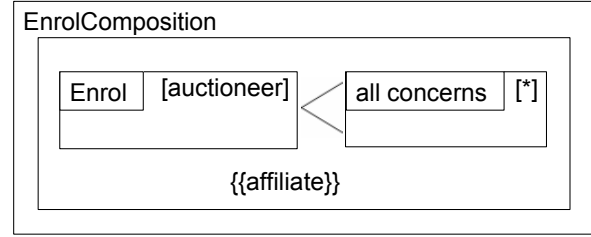


Figure 5: Composition of ‘Enrol’ concern

The visual representation is further elaborated in Figure 6. As discussed in Figure 3 and section 2.2 earlier, any requirement that the seller ends/cancels the auction (shown in Figure 6 with any requirement that has subject *seller* [S:| seller], object *auction* [O:| auction], and relationship *end* [R:| end] between them will be realized only if the auction has not begun ([S:| seller] [S:| auction] [R:| begin]). Here the constraint operator is $\{\{begin/end\}\}$. We have also used the visual representation of the *ifnot* RDL conditional base operator which has not been previously presented in section 2.

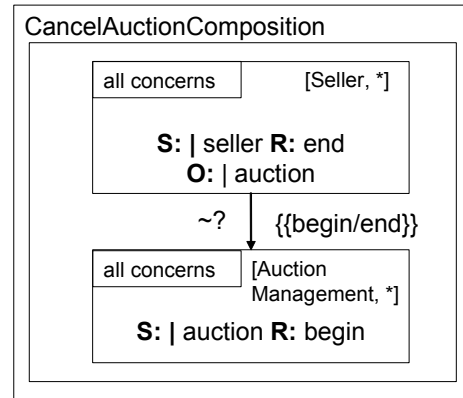


Figure 6: Composition ‘CancelAuctionComposition’

As shown in Figure 6, concerns and their subjects, object, and relationships identified by RDL can be represented in our notation in a compact and easy to perceive manner. This also highlights the scalability support of this visual representation. Moreover, this notation facilitates reusability by delimiting preceding subjects and objects with a ‘|’ sign. This enables the subject and object to be replaced by any other systems subjects and objects. Thus, the composition can be reused for any other system where subjects/objects differ but the rest of the composition semantics are unchanged.

4. Traceability

We have discussed before that our RDL is used to semi-formally represent any textual requirements, highlighting the syntactic and semantic elements of

particular significance within them. However, the RDL representation of these requirements and, in particular, their compositions is not very intuitive to an average end-user. Consequently we have developed a simple visual notation so that this visual representation of requirements and their compositions (i.e. mutual influences) can then be taken back to the stakeholders for validation. This process is illustrated in Figure 7.

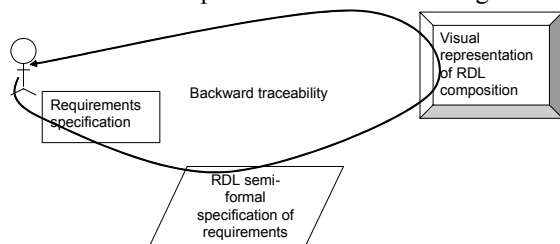


Figure 7: Backward traceability

We have also discussed that our visual notation provides corresponding elements for each composition element of the RDL and lists the sources of each of the composition participants (e.g., Figures 5 and 6). Thus, simply by establishing this process (presented in Figure 7) we have established a clear backward traceability link from the textual requirements to the RDL, visual representation and back to the source.

The dependency trace from the early requirements provided in the textual format to both their independent and composed semi-structured RDL specification helps the architects and designers in understanding the consequences of requirements dependencies, but it is not necessarily useable for communication with the stakeholders. Instead, the stakeholders use the visual representations of the compositions to verify whether the software engineer has correctly structured the requirements and composition between these requirements.

In case a question arises about a given composition, the backward traceability links are used to identify the sources of the composed concerns, which can then help resolve the questions. For instance, in case a modification needs to be incorporated into a software system, a set of impact analysis scenarios derived from the existing compositions can be prepared. These scenarios can then be presented for selection and validation to the relevant end-users the list of which is obtained from the visual notations.

Thus, through the visualisation and backward traceability links our approach promotes better understanding of requirements influences by both requirements analysts and end-users, as well as supports validation of compositions by the stakeholders.

5. Related Work

There has been very little previous work on requirements composition visualisation. However there has been some work carried out on requirements composition, and significant research on traceability and visualisation of requirements. Thus, we discuss the related work for each of these areas.

As mentioned before, work on requirement composition is particularly relevant for the AORE approaches [2, 11], and our work has evolved from addressing the shortcomings of these. Both [2, 11] have a relatively simple set of composition operators (where operators in base are called *actions* and in constraint *operators*) which have been identified from completed case studies. We provide a methodical approach to such operator derivation. Besides, our composition specification has much richer semantics based on the semantics of the requirements themselves and their dependencies. In addition, the need for composition specification in [2, 11] is rooted only in the judgement of the requirements engineer, whereas in our approach the need for compositions is identified from the *semantics* of requirements dependencies.

ThemeDoc [3] also has some elements of composition: by looking at the layers of themes on the *clipped action view* of the theme graph, we can determine some of the expected *sequence* of composition. The actual composition is then carried out at the design stage using the semantics of ThemeUML [3]. In our approach a more complete set of sequencing and conditional dependencies is identified, with composition being performed as part of RE analysis.

Some work is also currently underway in defining a composition mechanism for problem frames [12] and goal-based [13] approaches. Though these approaches themselves are not new (and well predate AORE), the issue of composition has not been vigorously addressed in them previously. Here again, it is the rooting of our work in semantics of requirements (i.e. the internal view) and use of NLP mechanisms that distinguishes our approach.

Some similarity to our work in modelling concerns along with their relationships can also be found with the approaches proposed in [14, 15]. In this work Sutton et. al. put forward a schema for concern classification where a section for relationships is also included. However, the issue of composition is not addressed.

Requirements traceability has long been acknowledged as an important issue in RE. It has also been known that different tasks within RE (e.g., release panning, change management, tracing to

design, etc.) have different perspectives and needs for tractability. Moreover, the needs of each individual company also widely vary with the kinds of traceability information required. Thus, most research in this area is fragmented per tasks. For instance, [16] looks at traceability links for software release planning; [17] looks at linking the requirements to their sources early on at the time of requirements elicitation task; [18] considers use of goal derivation graphs for traceably recording requirements derivation while the GOPCSD tool [19] generates goal models that are symbolically executed to trace the aspects from the root to terminal, etc. More recently some researchers have considered ways of integrating the fragmented task-based works on traceability into generic models [20, 21]. However, all these approaches focus on traceability links largely defined by *development activities* (e.g., release planning, etc.) investigating such links as cost-related, or added value links from one requirement to the other and refinement and change tracking links, etc. We, on the other hand, focus specifically on the (internal) *semantics of the requirements interdependencies* themselves, without addressing other (external process-related) links. We believe that it is necessary to understand the internal links first – these will form the foundation of the traceability framework, and can then be augmented with additional external links (if necessary). We also look at the traceability of *compositions* of requirements, not only individual requirements one by one, which is more usual for the other discussed approaches.

While there is a large body of work on visualization of UML-based AO design artefacts (e.g., [3, 22, 23]), the corresponding area for requirements visualisation is rather sparsely populated. A notable exception here is the Theme/Doc approach [3] that visually represents the requirements, which leads to the identification of crosscutting functionality. Yet, Theme/Doc does not cater for the visual representation of composition rules, which is the focus of our visualization approach.

[24] proposes an approach based on Interaction Pattern Specification (IPS) for visual representation of requirements and composition between the identified aspectual and non-aspectual requirements by composition operators (AND, OR, IN). The composed aspectual and non-aspectual requirements are visually represented in a composed sequence diagram. In comparison, our RDL-based visual notation has richer composition semantics, represented in a simple notation that is understandable and verifiable by stakeholder.

[10] has provided a visual representation for very simple composition of functional and crosscutting

requirements applying use case diagrams and three (<<overlap>>, <<override>>, <<wrapby>>) composition operator stereotypes. This approach is only applicable for use case based AORE models and a very limited set of composition semantics is addressed.

The FRIDA model [23] provides traceability from requirements to design phase, also, visually modelling the aspects, pointcut, and advices by stereotypes. But this visual representation is closer to UML based design which is not easily understandable by end-users and not easily verifiable.

In summary, compared to the other composition visualization approaches, our work has markedly richer composition semantics, based on the requirements dependency semantics themselves [5]. Our work also focuses on such semantics in building traceability links and compositions.

6. Summary and Conclusions

The main contribution of this paper is presentation of an RDL whose development has been motivated and grounded in the semantics and grammar of natural language requirements. Due to its roots, the RDL is as generic and flexible as the grammar of the natural language itself.

We have also discussed how this RDL can be used to reveal the mutual influences of requirements, using the composition operators derived from the very semantics of the requirements dependencies.

In the future we will continue the work on our RDL and, in particular, will validate its features through larger case studies.

We have also discussed how the RDL has been complemented with a simple visual notation for composition representation. This is particularly useful for validation of the requirements composition with non-technically minded stakeholders who may find understanding RDL-specified composition difficult.

Our future visualization work will focus on establishing automated traceability links (via a visualisation tool) between composition points of concerns picked out via semantics-based pointcuts; visual verification of the correctness of such pointcuts (with user participation), as well as a experiments to verify the scalability and reusability features of the composition representation.

7. 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).

8. References

- [1] R. Chitchyan, A. Rashid, P. Sawyer, A. Garcia, M. Pinto, J. Bakker, B. Tekinerdogan, S. Clarke, and A. Jackson, "Survey of (Aspect-Oriented) Analysis and Design Approaches," Lancaster University, Lancaster, Survey Report AOSD-Europe-ULANC-9, 2005.
- [2] A. Rashid, A. Moreira, and J. Araujo, "Modularisation and Composition of Aspectual Requirements," presented at 2nd International Conference on Aspect-Oriented Software Development, Boston, Massachusetts, 2003.
- [3] S. Clarke and E. Baniassad, *Aspect-Oriented Analysis and Design: the Theme Approach*. Addison-Wesley, 2005.
- [4] M. Jackson, *Problem Frames: Analyzing and Structuring Software Development Problems*. Addison-Wesley, 2001.
- [5] R. Chitchyan and A. Rashid, "Tracing Requirements Interdependency Semantics," submitted to Workshop on Early Aspects (to be held with ASOD 06), Bonn, Germany 2006.
- [6] R. M. W. Dixon, *A Semantic Approach to English Grammar*, 2 ed.: Oxford University Press, 2005.
- [7] Web Site: *Auction System Problem Description*, <http://lgl.epfl.ch/research/fondue/case-studies/auction/problem-description.html>, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, 2005.
- [8] I. Sommerville, *Software Engineering*, 7 ed: Addison-Wesley, 2004.
- [9] C. Chavez and C. Lucena, "A Metamodel for Aspect-Oriented Modeling," in *Workshop on Aspect-Oriented Modeling with UML (AOSD-2002)*. Enschede, The Netherlands, 2002.
- [10] J. Araujo, A. Moreira, I. Brito, and A. Rashid, "Aspect-Oriented Requirements with UML," Workshop on Early Aspects (held with AOSD).
- [11] A. Moreira, J. Araujo, and A. Rashid, "Multi-Dimensional Separation of Concerns in Requirements Engineering," presented at Requirements Engineering Conference (RE 05), Paris, France, 2005.
- [12] C. Haley, R. Laney, and B. Nuseibeh, "Deriving Security Requirements from Crosscutting Threat Descriptions," Aspect-Oriented Software Development Conf. (AOSD), Lancaster, UK, 2004.
- [13] Y. Yu, J. C. S. d. P. Leite, and J. Mylopoulos, "From Goals to Aspects: Discovering Aspects from Requirements Goal Models," presented at International Conference on Requirements Engineering, Kyoto, Japan, 2004.
- [14] S. M. Sutton and I. Rouvellou, "Modeling of Software Concerns in Cosmos," presented at International Conference on Aspect-Oriented Software Development, 2002.
- [15] S. M. Sutton and I. Rouvellou, "Concern Modeling for Aspect-Oriented Software Development," in *Aspect-Oriented Software Development*, R. E. Filman, T. Elrad, S. Clarke, and M. Aksit, Eds.: Addison-Wesley, 2004, pp. 479-505.
- [16] P. Carlshamre, "Release Planning in Market-Driven Software Product Development: Provoking an Understanding," *Requirements Engineering Journal*, vol. 7, pp. 139-151, 2002.
- [17] O. Gotel and A. Finkelstein, "Extended Requirements Traceability: Results of an Industrial Case Study," presented at 3rd International Symposium on Requirements Engineering (RE '97), Annapolis, Maryland, USA, January 6-10 1997.
- [18] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*: Kluwer Academic Publishers, 2000.
- [19] I. A. M. El-Maddah and T. S. E. Maibaum, "Tracing Aspects in Goal driven Requirements of Process Control Systems," in *Workshop on Early Aspects (held with AOSD)*. Lancaster, UK, 2004.
- [20] B. Ramesh and M. Jarke, "Towards Reference Models for Requirements Traceability," *IEEE Transactions on Software Engineering*, vol. 37, 2001.
- [21] A. Dahlstedt and A. Persson, "Requirements Interdependencies - Moulding the State of Research into a Research Agenda," presented at The Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 2003), held in conjunction with CAiSE 2003, Velden, Austria, 2003.
- [22] D. Stein, S. Hanenberg, and R. Unland, "A UML-based Aspect-Oriented Design Notation For AspectJ," Aspect-Oriented Software Development (AOSD 2002), Enschede, The Netherlands, 2002.
- [23] S. d. C. Bertagnolli and M. L. B. Lisboa, "The FRIDA Model," presented at Workshop on Analysis of Aspect-Oriented Software (held with ECOOP 2003), Darmstadt, Germany, 2003.
- [24] J. Whittle, J. Araujo, and D.-K. Kim, "Modeling and Validating Interaction Aspects in UML," presented at AOSD Modeling With UML Workshop (located with UML 2003), San Francisco, USA, 2003.