

Tracing Requirements Interdependency Semantics

Ruzanna Chitchyan, Awais Rashid

*Computing Department, Lancaster University, Lancaster
r.chitchyan@lancaster.co.uk, marash@comp.lancs.ac.uk*

Abstract

The problem of requirements inter-dependencies resulting in conflicts and trade-offs is long known in the requirements engineering community. In this paper we discuss how such dependencies can be deduced from the semantics of requirements and how their traceability can be maintained when concerns are formed from initially unstructured requirements, or requirements are composed for analysis purposes.

1. Introduction

A significant body of work has been devoted to the problem of requirements dependencies [1-6]. For instance, [1] proposes a classification of dependency types, which are then mapped to respective traceability links; [2] presents a survey of interdependency types identified from an industrial survey; [3, 6] summarize literature on interdependency work with a view to developing a general metamodel for all identified dependency types.

However, to our knowledge, none of the works in this area has addressed requirements interdependencies as originating from the *semantics* of the requirements themselves. Instead, the research has attended to detecting dependencies that are noticed when requirements are used in the software development process (e.g. during release planning [2], requirements and change management [1], reuse [7], etc.).

Unlike the other work in this area, our approach suggests that the language used for expressing requirements already reveals what dependencies should be expected between them. Consequently, rather than searching for *external* ways of dependency discovery (e.g. by observing dependencies through use of requirements), we look *into* the requirements themselves. In other words, we study the requirements' semantics, which exist prior

and independently of how these requirements are used in any particular case. We utilize the vast work on Natural Language Processing (NLP) that has studied the meaning of words and their link to the roles that these words play within natural language statements. We believe that, since most requirements are expressed in natural (sometimes semi-structured) language¹, the same work on NLP is also applicable to the requirements engineering domain.

This paper discusses our approach to utilizing NLP work in detection and analysis of requirements dependencies. The analysis part of our approach builds on the previous work on Aspect-Oriented Requirements Engineering (AORE) presented in [8, 9]. In particular, the work presented in this paper can be used to extend, refine and complement the composition semantics presented in [8, 9]. Like [8, 9] and [10], we specify composition using a set of composition operators. However, in all the previously developed AORE work, including [8-10], there are no means of tracing as to where the composition operators in each composition specification have come from. In the present paper we address this issue in particular.

The rest of this paper is structured as follows: section 2 discusses the background work related to concern composition and use of NLP in our approach. Section 3 presents our approach, using a small example to demonstrate the discussed concepts. Section 4 presents related work. The paper is concluded in section 5.

2. Requirements Composition and NLP.

The background work related to our approach falls into two areas: the previous work on Aspect-Oriented Requirements Engineering and NLP. Each

¹ Of course, some requirements are expressed in alternative ways, e.g., drawings, mathematical specifications, etc. We do not consider these types of requirements in this paper.

of these is separately discussed in a dedicated subsection.

2.1. Requirements Composition

Several AORE approaches, e.g. [8, 9] and [10] focus, on addressing the issue of separating aspectual requirements and their composition by defining composition rules.

In these approaches composition is used for ensuring requirements consistency [8, 9] or reducing redundancy of repeating requirements [10] (e.g., error handling scenarios). In [8, 9] potential conflicts are detected when influences from several compositions are found to affect a given (part of) requirement. An example of such influences is, for instance, contradictory changes to one requirement imposed by several others.

Both [8, 9] use XML for artifact representation and composition rule specification, thus, keeping

artifacts and compositions structured, semi-formalized, yet simple and understandable.

In order to illustrate the composition rules of [8, 9], we use an extract of requirements for ATM and compatibility issues, from requirements for the Toll Gate System presented in [8]: “..The registration of authorized vehicles includes the owner’s personal data, bank account number and vehicle details. The gizmo is sent to the client to be activated using an ATM that informs the system upon gizmo activation...”

From this extract [8] defines two units of requirements modularization with their corresponding requirements and sub-requirements: ATM and Compatibility, presented in Figure 1.

As illustrated in Figure 1, all the requirements and sub-requirements are given unique identification numbers. These identification numbers are used to refer to the specific requirements during composition. Such example of composition is presented in Figure 2.

ATM:

Requirement id="1": **The ATM sends the customer's card number, account number and gizmo identifier to the system for activation and reactivation.**

Requirement id="1.1" **The ATM is notified if the activation or reactivation was successful or not.**

Requirement id="1.1.1": **In case of unsuccessful activation or reactivation the ATM is notified of the reasons for failure.**

(a)

Compatibility:

Requirement id="1": **The system must be compatible with systems used to:**

Requirement id="1.1": **activate and reactivate gizmos;**

Requirement id="1.2": **deal with infraction incidents;**

Requirement id="1.3": **charge for usage.**

(b)

Figure 1: AORE [8, 9] elements: (a) ATM requirements module, (b) Compatibility requirements module. (Figure adapted from [8])

The composition rule² depicted in Figure 2 states that the requirement with id 1.1 in Compatibility module is imposing a constraint (the Constraint tag in Figure 2) on all requirements (shown with id="all") of the ATM module. The imposed constraint contains two operators, where action-operator³ (referred to as *action* from now on) defines the *type of relationship* (i.e. *dependency*) between the constraining and constrained

requirements; and condition-operator⁴ defines how these relationships should be enacted.

```
<Composition>
  <Requirement "Compatibility" id="1.1">
    <Constraint action-operator="ensure"
      condition-operator="with">
      <Requirement "ATM" id="all"/>
    </Constraint>
  </Requirement>
</Composition>
```

Figure 2: An example of composition. (Figure adapted from [8]).

² In order to keep the discussion simple and relatively general, we do not use the detailed artifact names, and types, etc. specific to each of the discussed approaches. Interested reader is referred to [8] and [9].

³ Called *action* in [8] and [9].

⁴ Called *operator* in [8] and [9].

In other words, the composition rule in Figure 2 states “Compatibility Requirement 1.1 (that the system must be compatible with the systems used to activate and reactivate a gizmo) must be ensured with regards to all Requirements of ATM” [8].

Thus, the composition in [8, 9] provides us with a clear picture of dependencies between requirements of different modules (i.e. ATM and Compatibility in our example): *action* defines the type of activity that the dependency implies; *condition-operator* define the *sequence or condition* of the dependencies.

In addition, a third operator can be used to define the (additional) effects of the enacted dependencies on the outcome of the composition.

Yet, there is no systematic way of operator derivation or their application to specific compositions. Up till now these have been derived from case studies [8, 9] and applied in particular compositions at discretion of the requirements analyst.

However, we maintain that the choice of a particular *operator* in a composition cannot be arbitrary. Such a choice has to be defined by the *semantics of the dependencies* that exist between the requirements of modules that are being composed. In order to reveal the semantics of such dependencies, we apply the work on NLP discussed in the following subsection.

In the present paper we focus on the derivation of semantics of operators that we called *actions*.

2.2. NLP

Several prominent works in linguistics [11-13] have shown that there is a clear link between the meaning of

the words and their grammatical behavior. Such a link can be illustrated via a simple experiment presented in [13]: two English speakers were asked about the correct use of an archaic English verb *gally* which was used in whaling. They were presented with a sentence “Sailors galled the whales.” Then they were asked if use of *gally* in the sentence “Whales *gally* easily.” is appropriate. The speaker who thought that *gally* meant *see*, believed that it was incorrect, as we don’t say “Sailors saw the whales. *Whales see easily.*” On the other hand, the speaker who thought *gally* to mean *frighten*, believed that it was correct to say “Whales *gally* easily, as it is correct to say *Whales frighten easily*”.

In line with the above experiment, Dixon [12] suggests that the varying grammatical behavior of verbs is the result of the differences in their meaning. Thus, using this principle, [12] groups verbs into several semantic categories. The verbs in each semantic category require the same type of participating roles. For instance, all *Giving* type verbs require a *Donor*, *Gift*, and *Recipient* roles, as in *Allan (Donor) gave the keys (Gift) to Peter (Recipient)*; all *Attention* verbs require a *Perceiver* and an *Impression* role, as in *The instructor (Perceiver) witnessed the accident (Impression)*, etc. In some cases certain roles can be omitted, or understood from the context or from most common use of the verb.

As illustrated in Figure 3, all in all 19 first level (63 in total) verb groups are identified in Dixon’s verb classification [12].

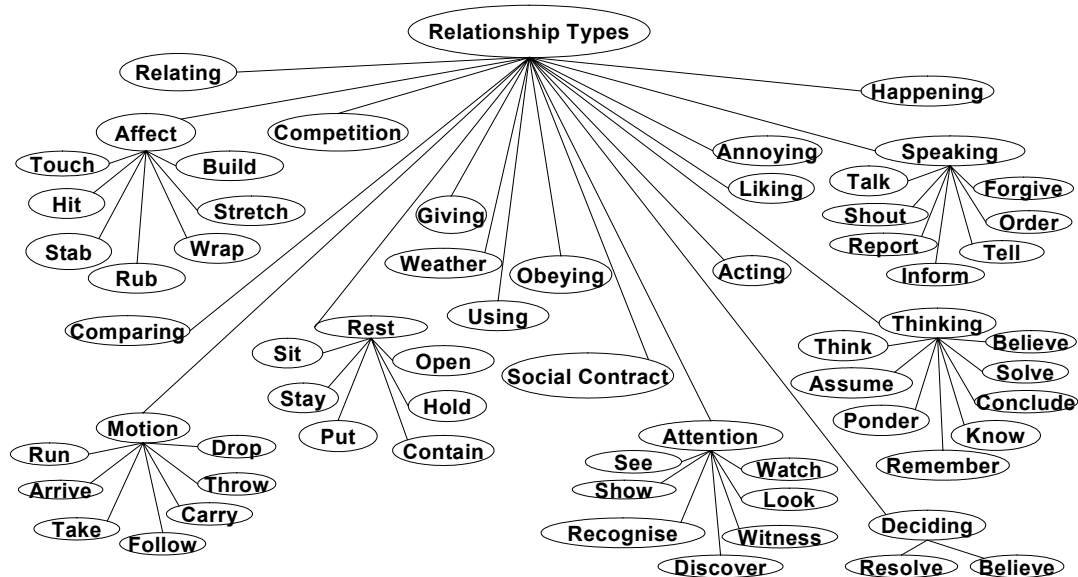


Figure 3: Verb categories by Dixon.

In our work we use the semantic categories of [12] as basis for identifying the types of relationships between concerns and, eventually, deriving composition operators.

3. Requirements Interdependency Semantics

We observe, that generally in natural language, the semantics of action-type dependencies (denoted by *action-operators*, or *actions* as per [8, 9]) are expressed by verbs or verb phrases. But, in accordance with Dixon's verb classification [12] there is only a limited set of broad meanings of verbs, thus, there must be only a limited set of broad dependency types (and, correspondingly, operators of *action* type) that correspond to the verb categories.

3.1. An Alternative Verb Categorization

As illustrated in Figure 3, there are 63 verb classes suggested by Dixon [12], however, often the verbs grouped in one class can have either positive or negative sense, e.g., both *utilize* and *waste* belong to the *Use* category; *increase* and *decrease* belong to the *Change* category, etc. Thus, if we were to associate an action for positive and negative sense to each of the verb categories of [12], we would end up with an overwhelming number of actions, making our composition semantics too complicated.

We also must review the suitability of the semantic categories in [12] from the perspective of composition semantics. In doing this, we discover two major issues:

- 1) Since at composition time we are interested in more general semantics of actions, semantics of some sub-groups in [12] are too fine grained for our needs. For instance, the semantics of Touch and Hit sub-groups differ only in the *intensity* of contact action; Look and Watch differ only in the *duration* of the looking action, etc.
- 2) Some groups that have close general semantics are separated. For instance Giving verbs comprise a separate category, however, very often Take verbs (e.g. take/bring/fetch for/to Ann), similar to Giving, also mean moving the possession of something to someone/ something else.

Using the above two observations, we have re-arranged Dixon's categories [12] to fit our purpose of composition semantics. The reduced set of verb classes for use with composition is presented in Figure 4.

Let us now discuss the major differences between our (Figure 4) and Dixon's [12] (Figure 3) classifications.

From Figure 4 one can see that some major groups of Dixon's categories are preserved (Affect, Move, Rest), while some others seem to have been completely removed (e.g. Attention, Thinking, Speaking, etc.). The missing Dixon's categories have, in fact, been amalgamated or positioned within another tree structure. Thus, for instance, the Attention and Thinking categories have been combined into the Metal Action category, which also includes the Liking and Annoying groups. The Mental Action category now collects all verbs related to thinking and attention (which also includes a strong mental element), as well as mentally and emotionally motivated liking and annoying verbs. These verbs are placed together as, from the requirements interdependencies perspective, there is no significant difference between attention-related or thinking related effects of requirements.

The Speaking Dixon category has been replaced by Communicate category, which (in addition to amalgamating some sub-groups of Speaking into one) also includes the Show sub-category which in Dixon's classification belongs to the Attention group. Show has been moved to the Communicate category since it too involves some communication of knowledge from one person to the other.

The General Action category acts as a first level category for a number of groups related to more general behavior that has not been already included in other groups.

It should be noted, that though we have re-arranged Dixon's classes, in most instances (except for the General Actions case) the newly formed categories still maintain the general notions of participatory roles. Thus, the Mental Action category which includes the Thinking category (with Coagulator and Thought roles) and Attention category (with Perceiver and Impression roles) still requires two roles as the Coagulator in many ways is similar to Perceiver who perceives the Thought as an impression. While Dixon's classification distinguishes perception via eyes, ears, etc. and mental perception, our categorization does not. Similarly, the Experiencer and Stimulus in Liking and Annoying groups of Dixon are also mapped to Perceiver and Impression.

In case of General Actions category, each of the sub-groups has its own set of participating roles, as the top level category is merely a convenience group, and not a single coherent semantic category.

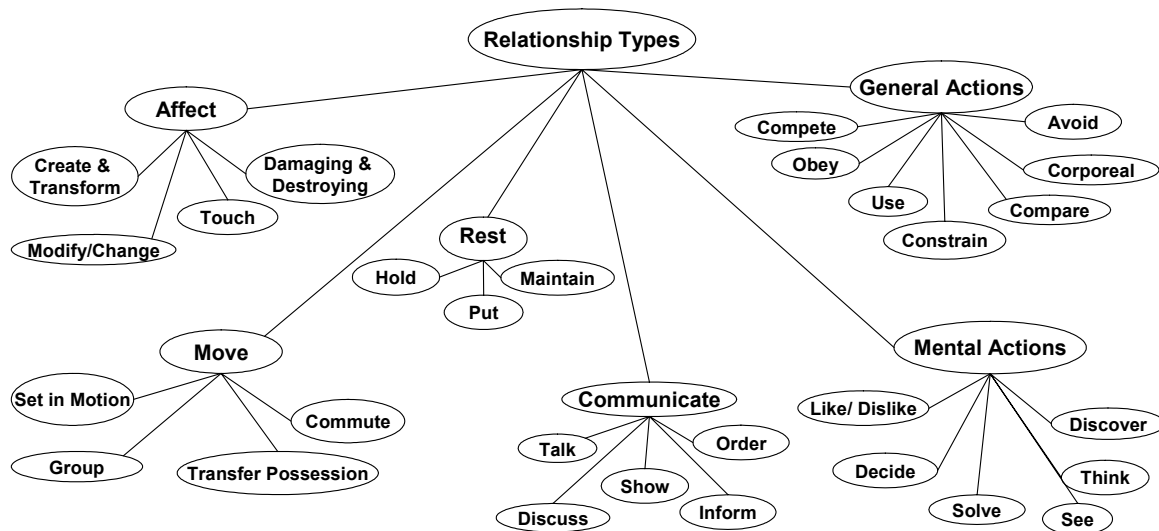


Figure 4: Reduced Relationship Set

3.2. Deriving Composition Operators from Verb Categories

In the above sub-section we have discussed how we have categorized the verbs into semantically related groups from the requirements dependencies perspective. Now, by identifying that a verb in a requirement belongs to a particular category, we are essentially stating that it carries certain semantic and syntactic features. Thus, we can now derive representative composition operators for actions from each group that, when used in composition, will indicate the semantic and syntactic properties of the verb and its related roles.

Since in some categories the positive and negative semantics of the verbs need to be indicated, one category may have 2 operators. Figure 5 presents the initial set of the action-operators derived from categories of Figure 4. For clarity of presentation, the operators in Figure 5 are depicted in *italics*.

3.3. Tracing Requirements Interdependency Semantics to Composition

In the above sections, we have discussed that the requirement compositions must be justified by the semantics of the requirement dependencies.

We have also derived a set of *action-operators* that should reflect the general types of such dependencies. As a result, we should be able to ensure that the requirement compositions are semantically justified,

rather than arbitrarily provided by a requirements analyst.

For instance, the ATM concern (presented in Figure 1) clearly demonstrates a “Transfer Possession” activity (send the card... for activation) which implies the “provide” action-operator, as well as a “Communicate” category with “inform” operator (“... ATM is notified...”). It is also obvious what other concerns participate in such relationships, as they take on specific roles customary to the corresponding activities used within the respective requirements: thus, the [toll] system participates in the above relationships with the ATM, as it is explicitly stated in the *Recipient* role for the requirement with id 1, and is implied in its sub-requirements (with ids 1.1 and 1.1.1).

Of course, the identification of such relationships between requirements or concerns does not immediately necessitate a composition specification. Rather, it provides a sound basis and readily available composition operators and identified participants in the composition, should such composition specification be needed⁵.

For instance, the composition example presented in Figure 2 can be observed from the Compatibility concern and ATM concern depicted in Figure 1. The Compatibility requirement that “the system must be

⁵ Whether or not such a composition is needed can be decided, either using the conflict identification and resolution approach in [8, 9], or a set of guidelines, etc. This issue is not discussed further in the present paper.

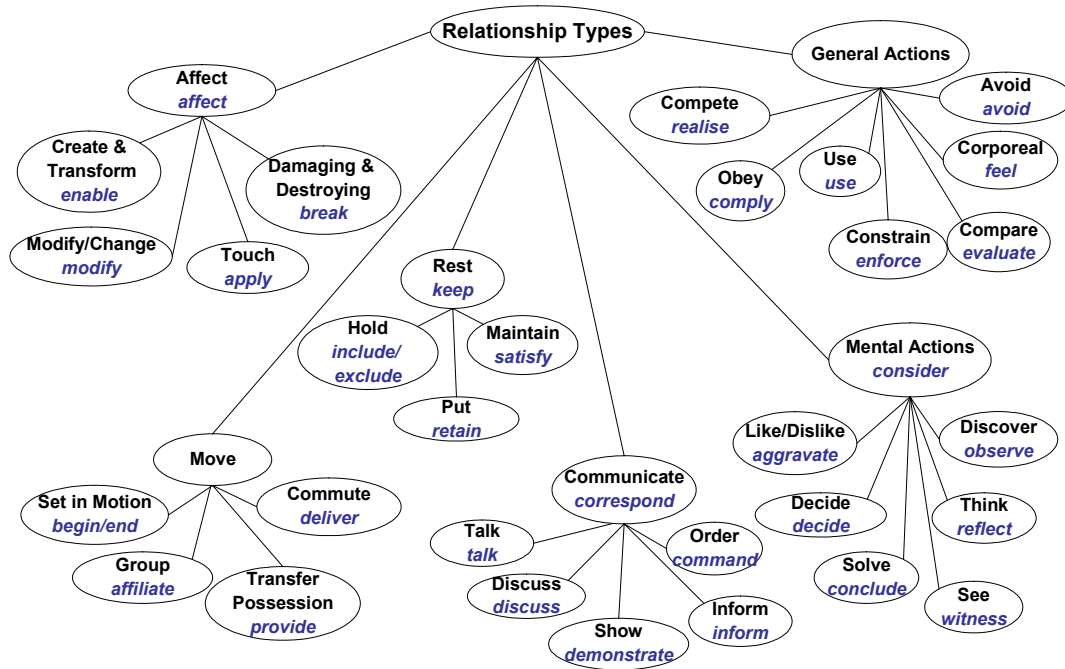


Figure 5: Composition Operators.

compatible with systems used to activate and reactivate gizmos...” reveals the composition action-operator “enforce” (note that we have called this operator differently from that in [8]) of *General Action* group’s *Constraint* sub-group with the “system used to activate and reactivate gizmo...” being an object of the constraint. Since all requirements within ATM relate to such a “...system used to activate and reactivate gizmo...”, they all participate in the composition.

It should be noted, that our view of composition extends that of the [8, 9] (for instance we use the notion of “semantic-based composition points”, etc.). More details on our approach to composition are provided in [14].

Thus, the composition derivation and the semantics of its participating elements (e.g. actions, specific requirements, etc.) can be directly and transparently traced to the semantics of individual requirements and concerns involved in the composition and their interdependencies. This is briefly illustrated in Figure 6: the dependencies between requirements of two concerns are mapped to an Operator; when composition is specified, clear references to the specific requirements within the concerns and their dependency (reflected by the operator) are clearly preserved.

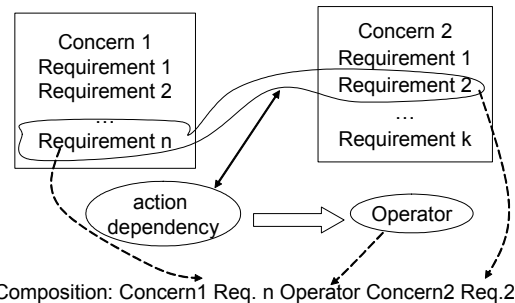


Figure 6: Traceability to Composition

4. Related Work

Work on requirements interdependencies has focused on identifying and recording links between requirements, and so has been carried out under the umbrella of more general work on requirements traceability [6]. Within the traceability scope, most research has focused on specific software development tasks, looking at what dependencies matter for requirements release planning, reuse, change management, testing, and so on.

For instance, [2] describes a set of dependencies from the perspective of release planning found through an industrial survey. The set of dependencies identified in [2] are AND (R_1 requires R_2 and R_2 requires R_1), Requires (R_1 requires R_2 but not R_2 requires R_1),

Temporal (R_1 has to be implemented before/after R_2), CValue (R_1 affects the value of R_2 for a customer), ICost (R_1 affects the cost of implementing R_2), and OR (only one of $\{R_1, R_2\}$ needs to be implemented). Thus, like [4], it is solely aimed at identifying which requirements should be selected for implementation.

In [15] the focus is on establishing links between the requirements and their sources early on. In [16] use of goal derivation graphs for traceably recording requirements is considered.

More recently work is underway to provide an integrated framework for traceability [1] and dependency [6] links identified in task-based approaches. For instance, [6] distinguishes three main dependency types: Structural, Constrain, and Cost/Value. Each of these types has several sub-types. Thus, Structural type contains Refined_to, Change_to, Similar_to, and Requires sub-types; Constrain types contains Requires (same as in Structural group) and Conflicts-with; and finally Cost/Value group contains Increase/Decrease cost and Increase/Decrease value sub-types.

All the above discussed task-based approaches focus on classification of interdependencies to sets of traceability link types defined by development activities (e.g. release planning, etc.) which are not defined by (i.e. are external to) the requirements themselves. We, on the other hand, focus on the *semantics of the interdependencies* which are *internal* to the requirements. We are of the view that it is such semantics that should be used as basis for consequent systematic determination of more abstract types of traceability links.

On the other hand, unlike the unification work [1, 6], our approach does not claim to cover the whole spectrum of *all* possible types of dependency links, as *external* task-specific links may not always be evident from semantic analysis of requirements. Thus, in this respect, our work complements that of [1, 6].

A few other approaches have considered the semantics of requirements as source of their dependencies. For instance, [7] not only identifies several generic development-related dependency types (*Refinement*, *Dependency*, and *Representation*), but also provides more specific sub-types of Dependency (e.g. *monitors*). These specific sub-types are arbitrarily derived by an analyst using his/her domain knowledge and the textual description of requirements.

We have already mentioned that some work on Aspect-Oriented RE (e.g., [8, 9]) considers dependencies from the perspective of composition: their composition *actions* (e.g., provide, enforce, affect, etc.) are derived, similar to [7], using the

analyst's domain knowledge, the contribution tables, and the textual description of requirements.

This last set of approaches can largely benefit from our work, as we provide a *systematic way* of deriving semantic dependency types that are rich yet reusable for all kinds of requirements.

Our work also bears some resemblance to ThemeDoc [17, 18] in use of verbs. However, in ThemeDoc the *action verbs* are provided manually (i.e. externally) by a requirements engineer, while in our case the *actions* depict the *internal* semantics of the verb phrase of the requirement. Besides, ThemeDoc uses *action verbs* as a way of crosscutting concern identification, while we focus on dependency type identification between concerns and their requirements. We use such dependencies for composition specification of requirements, while ThemeDoc does not deal with composition at the requirements level, deferring it to the design stage to be supported via ThemeUML [18]. Thus, similar to [8, 9] ThemeDoc too could utilize our work in its identification of types of *action verbs* and further.

5. Summary and Conclusions

In this paper we have presented an NLP-based approach that assists in identification of various kinds of semantically motivated dependencies between requirements. While this work can be used simply as a way of classification of such dependencies, contributing to the work on traceability and dependency classification, it also contributes significantly to the requirements composition in AORE.

In summary, with respect to AORE, this work contributes by:

- enriching composition with the semantics of the relationships between requirements;
- providing a systematic way of deriving composition operators from the semantics of requirements relationships.

In order to keep the material of this paper focused, as well as due to space limitations, we have limited the discussion only to the action-based (i.e. verb and verb phrases- based) dependency derivations. A wider view of our work, including temporal and conditional dependencies and degree (i.e.) strength or requirements relationships is outlined in [19], [14], and [20].

As part of our future work, we will further improve our approach, and integrate it more closely with the composition and trade-off resolution work of [8, 9], as well as validate it through a larger industrial case study.

6. 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).

7. References

- [1] B. Ramesh and M. Jarke, "Towards Reference Models for Requirements Traceability," *IEEE Transactions on Software Engineering*, vol. 37, 2001.
- [2] P. Carlshamre, K. Sandhal, M. Lindvall, B. Regnell, and J. N. o. Dag, "An Industrial Survey of Requirements Interdependencies in Software Product Release Planning," 2001.
- [3] 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.
- [4] J. Karlsson, C. Wohlin, and B. Regnell, "An Evaluation of Methods for Prioritizing Software Requirements," *Information and Software Technology*, vol. 39, pp. 939-947, 1997.
- [5] J. Karlsson, S. Olsson, and K. Ryan, "Improved Practical Support for Large-scale Requirements Prioritisation," *Requirements Engineering Journal*, vol. 2, pp. 51-60, 1997.
- [6] A. G. Dahlstedt, "Requirements Interdependencies - Towards an Understanding of their Nature and Context of Use," Department of Computer and Systems Science, Stockholm University, Stockholm, Licentiate Thesis SE-164 40 KISTA, 2004.
- [7] A. v. Knethen, B. Paech, F. Kiedaisch, and F. Houdek, "Systematic Requirements recycling through Abstraction and Traceability," presented at Requirements Engineering, Essen, Germany, 2002.
- [8] 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.
- [9] 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.
- [10] J. Whittle and J. Araujo, "Scenario Modeling with Aspects," *IEE Proceedings - Software*, vol. 151, pp. 157-172, 2004.
- [11] B. Levin, *English verb classes and alternations: a preliminary investigation*. Chicago: The University of Chicago Press, 1993.
- [12] R. M. W. Dixon, *A Semantic Approach to English Grammar*, 2 ed. Oxford: Oxford University Press, 2005.
- [13] K. L. Hale and S. J. Keyser, "A View from the Middle," MIT, Cambridge, MA, Center for Cognitive Science 1987.
- [14] R. Chitchyan, S. S. Khan, and A. Rashid, "Modelling and Tracing Composition Semantics in Requirements," in *submitted to Early Aspects Workshop (to be held with AOSD 2006)*. Bonn, Germany, 2005.
- [15] O. Gotel and A. Finkelstein, "An Analysis of the Requirements Traceability Problem," presented at 1st International Conference on Requirements Engineering (ICRE '94), Colorado Springs, Colorado, USA, 1994.
- [16] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-Functional Requirements in Software Engineering*: Kluwer Academic Publishers, 2000.
- [17] E. Baniassad and S. Clarke, "Theme: An Approach for Aspect-Oriented Analysis and Design," presented at International Conference on Software Engineering, Edinburgh, Scotland, UK, 2004.
- [18] S. Clarke and E. Baniassad, *Aspect-Oriented Analysis and Design: the Theme Approach*: Addison-Wesley, 2005.
- [19] R. Chitchyan, I. Sommerville, and A. Rashid, "CoCA: A Composition-Centric Approach to Requirements Engineering," presented at International Conference of Requirements Engineering, Paris, France, 2005.
- [20] R. Chitchyan, A. Sampaio, A. Rashid, P. Sawyer, S. Khan, "Initial Version of Aspect-Oriented Requirements Engineering Model", Lancaster University, AOSD-Europe project report (D36) No: AOSD-Europe-ULANC-17, February 2006.