Cross-Document Dependency Analysis for System-of-System Integration

Syed Asad Naqvi[1,2], Ruzanna Chitchyan[1], Steffen Zschaler[1], Awais Rashid[1], and Mario Südholt[2]

[1] Lancaster University, Lancaster, UK
naqvis@comp.lancs.ac.uk, r.chitchyan@lancaster.ac.uk, szschaler@acm.org, awais@comp.lancs.ac.uk
[2] DCS, École des Mines de Nantes, Nantes, France
Mario.Sudholt@emn.fr

Abstract. Systems-of-systems are formed through integration of individual complex systems, often not designed to work together. A number of factors can make this integration very challenging which often leads to catastrophic failures. In this paper, we focus on three major classes of system-of-system integration problems: managerial independence, interface incompatibility, and component-system complexity. We then present an aspect-oriented requirements description language (RDL) which uses natural language analysis capabilities to reason about dependencies across the documentation of the constituent systems of a system-of-systems. The aspect-oriented compositions in the RDL also facilitate specification of cross-document constraints and inconsistency resolution strategies, which can be used for deriving proof obligations and test cases for verification and validation of the emergent behaviour of a system-of-systems. We showcase the capabilities of our RDL through a case study of a real-world emergency response system. Our analysis shows that the querying and composition capabilities of the RDL provide valuable support for reasoning across documentation of multiple systems and specifying suitable integration constraints.

1 Introduction

As software systems become increasingly pervasive in our daily lives, we are seeing the emergence of a new class of systems, that of, systems-of-systems (SoS) [1]. These SoS are at least an order of magnitude greater in complexity than their conventional counterparts. Examples of such systems are airport management systems, airline alliances, healthcare systems, disaster response and recovery systems, etc. However, for a SoS to function effectively, all the constituent systems need to work towards a common overall goal. This is not always the case given that a SoS comes into being as a consequence of emergent rather than pre-planned requirements. Even in case of pre-planned integration, unforeseen problems can arise owing to the dichotomy between individual system goals and those of the SoS. A recent example of such dichotomy can be observed in the

Heathrow Airport Terminal 5 problems, due to incompatibilities between the baggage handling system and the check-in system. The problem was further compounded by the fact that, as is usual in SoS, both constituent systems were under the control of different organisations—the goals of a SoS have to be understood and maintained across such organisational and system administration boundaries.

Reconciling the goals of a SoS with those of its constituent systems is non-trivial—a SoS is an ultra large-scale system [2] that comes into being by the collaboration of a number of systems that may belong to different domains. This multi-domain characteristic of SoS is often referred to as heterogeneity (or diversity) [3–7] and has been identified as one of the basic properties of SoS. Another common characteristic in SoS is distribution [1, 4, 5, 8–10] which entails geographical distribution of the collaborating systems forming part of a SoS. Furthermore, because of its scale, a SoS affects and is also directly affected by external forces such as political, economic, social, and legal factors.

The need for taking a holistic view of the SoS means that the stakeholders from the constituent systems need to communicate with each other and understand each other's perspectives. The heterogeneity and distribution inherent to a SoS make such communication extremely challenging. Furthermore, the influence of external factors needs to be understood when reasoning about the overall behaviour of a SoS.

The key challenge for stakeholders in a SoS is, therefore, to understand or learn about other systems (besides their own) and external influences [4, 11]. A large portion of this learning takes place by studying the various description and specification documents of other systems as well as legal guidelines, operational procedures, etc. Especially during the process of integrating different systems to form a SoS the system specifications, operational procedures, user manuals, business case documents, test reports, etc. of the different systems need to be analysed, perhaps cross-examined, to find conflicts and also opportunities for optimisation. The scale of this task, similar to the extreme complexity of a SoS, is beyond what can reasonably be achieved by a team of engineers without scalable automation support.

Substantial advances have been made in the field of natural language analysis in the past two decades that can be exploited to study large document sets. Similarly, with the recent emergence of aspect-oriented requirements engineering (AORE) techniques [12, 13], there are new reasoning mechanisms available to both study and specify constraints that crosscut multiple system specifications. In this paper, we present an aspect-oriented requirements description language (RDL) [14] and show how we utilise the semantics of the natural language itself to both explore and capture cross-document dependencies, as well as conflict resolution strategies, in RDL-based aspect-oriented composition specifications. Since the composition specifications are based on natural language semantics, they facilitate intentional reasoning about cross-document dependencies; that is, reasoning about the stakeholders' intentions as expressed in the document text. This allows us to better understand stakeholders' requirements for SoS

and identify and resolve any discrepancies between the overall goals of the SoS and those of its constituent systems.

The remainder of the paper is structured as follows. In Sect. 2 we discuss three main classes of SoS integration problems. Section 3 presents the natural language analysis concepts pertinent to our RDL, its semantic queries and semantics-based composition specifications which are, respectively, used to identify cross-document dependencies of interest and specify resolution policies. Section 4 details a case study from the emergency-related communications domain to show how the RDL can be used to tackle the three classes of integration problems highlighted in Section 2. We discuss some of the open research problems in Sect. 5 and how the RDL's capabilities may be extended to tackle these. Section 6 reviews related work and Sect. 7 concludes the paper.

2 Integration Challenges in Systems-of-Systems

The key factors underpinning the integration problems in SoS fall intro three main classes:

1. The collaborating systems in a SoS are managerially independent;
2. The evolution trajectory of the processes and interfaces of individual systems does not account for potential future collaboration or communication with other systems in a SoS context;
3. The individual systems are often, themselves, extremely complex and therefore difficult to understand and integrate with other systems.

We next discuss each of these classes in detail, with the help of some well-known examples of SoS integration failures.

2.1 Managerial Independence

Our notion of Managerial Independence is that different systems in a SoS are being managed as well as operated by different groups of people. Though managerial independence is a key characteristic of SoS [15], it also poses two major integration challenges:

– *Decision-making and maintenance of an overall vision at the SoS level:* The managers of individual systems are often unaware of the protocols and procedures of other systems so there is a lack of overall vision and control. Furthermore, the decisions required for integration and co-working have to be made by the cooperation and agreement of all the concerned parties. Obtaining this agreement can often be non-trivial even impossible. An example of this is the European Union. In June 2008 the people of the Republic of Ireland (which represent only one percent of the EU population) voted to reject the Treaty of Lisbon. For the treaty to come in to force, it requires ratification from all 27 countries in the EU. Even though 25 countries had ratified the treaty, rejection from just one country meant that the treaty could not come into effect. Such problems faced by decision-makers in the SoS context have also been highlighted in [4].

– *Managing a large-scale socio-technical system:* A SoS is inherently a socio-technical system that encompasses the activities of the humans as well as the hardware and software systems in its boundary. The individual systems that collaborate in a SoS will often continue to operate independently in order to provide some functionality independent of the SoS. Participating in a SoS puts additional stress on the human part of the individual systems. People may have to concentrate on two different lines of work. Concerns about errors arising from excessive workload and the interactions of various human roles in a SoS have also been put forward as a major issue [16].

An example of SoS integration failure due to Managerial Independence is the failure of Heathrow Terminal 5 (T5) on its opening day. T5 was built at a cost of 4.3 billion pounds. The period from project commencement to terminal opening day was approximately five and a half years. The last six months were an operations readiness period in which the final preparations before opening—staff training and terminal systems and process testing—were to take place [17].

The terminal fell seriously short of expectations on its opening day. Thousands of people faced a chaotic situation when the terminal's baggage handling system stopped working. Sixty eight flights were cancelled [17] and thousands of people were deprived of their luggage.

The examination of what went wrong on the opening day reveals that the cause of the problems was a systems integration and coordination problem between the two main agencies responsible for operating T5: the British Airport Authority (BAA), which is the agency that built the terminal, and British Airways (BA), the (only) airline operating from the terminal. Both of these agencies were working together and were responsible for the operations of the terminal. The lack of communication, coordination, and integration between BAA and BA prior to and on the opening day was the root cause of the following problems:

– Many amongst the BA staff were not familiar with the equipment that they were supposed to operate. This equipment had been provided by BAA [17, 18].
– BA claimed that they were unable to completely test the software systems under their control because BAA did not finish construction of the building in time [19].
– The baggage system failed because BAA had responsibility for the system at baggage check-in while BA had responsibility for the baggage loading part of the system [17]. The system finally shutdown because the rate of baggage check-in was higher than the rate of baggage loading [17].
– There was no crisis management system setup between BA and BAA at the terminal level [17].

Colin Matthews (chief executive of BAA) noted the lack of cooperation between the management of BA and BAA as the main cause of the problems at Terminal 5. In a statement [17] he said:

> "However well the airport operator and the airline operator, BA, are working it is also vital that the two are absolutely integrated and together. I think that during the construction of Terminal 5 that appeared

to be the case. Around about or just prior to the opening of T5 it seems that that togetherness deteriorated. It is that togetherness that allows you to cope with the issues that arise on the day."

2.2 Incompatible Interfaces

Systems evolve over the course of their lifetime to meet new demands. Often this evolution takes place in a stove-pipe-like narrow domain. The protocols and interfaces that the system defines are often not meant for communication, collaboration, or evolution beyond this specific domain. This lack of foresight in the systems' architecture places extra strain [20] when these systems are required to collaborate with other systems. There is also the principle of encapsulation [21] in systems engineering to consider. This principle calls for systems to be closed off from the outside and to hide their implementation and inner working. In order for individual systems in a SoS to collaborate, they may need to break the principle of encapsulation and allow invasive access [22] to their inner working from other systems [3].

The interoperability challenge [23] facing the various emergency response agencies in the US is an example of this class of problems. The emergency-response communication systems of various agencies and jurisdictions often evolve without taking into consideration the need to communicate with other agencies and jurisdictions.

An example of SoS integration failure, due to Incompatible Interfaces, is the chaotic rescue operation that took place after the Air Florida Flight 90 disaster in Washington DC [24]. On 13 January 1982 the Air Florida Flight 90 crashed into the Potomac River shortly after takeoff. A number of federal, state, and local emergency response agencies took part in the rescue effort. The rescue effort was greatly hampered due to the lack of compatibility between the communication systems of the different agencies. The different emergency-response agencies had evolved their communication systems independently without taking into consideration the requirements of integrated rescue efforts with other agencies. Consequently, the rescue personnel from different agencies could not communicate with each other [24].

2.3 Complexity of the Collaborating Systems

The complexity of a system has a direct impact on its understandability. Given that the constituent systems in a SoS are often highly complex themselves, integration problems arise because of one's inability to fully comprehend the workings of the individual systems. As a result, it is often not possible to predict the behaviour of the participating systems within the context of the SoS. As the participating systems in a SoS are often interdependent on each other, it may take only one system to malfunction for the entire SoS to break down. Such reliability problems in highly complex systems have been discussed in [4].

An example of SoS integration failure due to the complexity of collaborating systems is the loss of the Mars Climate Orbiter. The Mars Climate Orbiter

(MCO) started its journey to Mars on December 11, 1998 from Cape Canaveral, Florida. Its mission was to gather information about the Martian atmosphere and to act as a relay station for the Mars Polar Lander Mission. On September 23, 1999 it was lost while trying to setup an orbit around Mars. The MCO was designed, developed, and operated by the collaboration of two agencies—NASA and Lockheed Martin Astronautics (LMA) [25]. The cause of the crash was in a piece of software that was producing its results in English units instead of metric units. But according to Dr. Edward Weiler, NASA's Associate Administrator for Space Science:

> "The problem here was not the error, it was the failure of NASA's systems engineering, and the checks and balances in our processes to detect the error. That's why we lost the spacecraft." [26]

The review panel that conducted the analysis of the failure found that the best processes and standards for software development had been followed. As noted in [27], the problem was not that of methodology. The failure instead arose from the "sheer complexity" of the system . The different consituent systems of the space craft were so complex that errors within them could not be detected despite the use of rigorous means and best practices.

Of course, the above three classes of problem are not orthogonal. As shown in Table 1, all three SoS integration failure cases: T5, Air Florida Flight 90 and MCO, exhibit multiple classes of failures.

Class of Failure *SoS*	Managerial Independence	Incompatible Interfaces	Complexity of Collaborating Systems
Heathrow T5	√		√
Air Florida Flight 90	√	√	
Mars Climate Orbiter	√	√	√

Table 1. Multiple causes of SoS integration failures

In this paper, we propose natural-language document-processing techniques to help uncover the above classes of problems. Any such cross-document analysis must provide three capabilities:

- Querying over multiple documents in a fashion that accounts for diverse writing styles and usage of different terms to refer to the same concepts;
- Specification of constraints across multiple documents in order to explicitly capture conflict resolution strategies and, subsequently impose them, through derivation of proof obligations or suitable test cases for the SoS;
- Automation support to aid the engineers and stakeholders in such querying and constraint specification during SoS integration.

In the next section, we discuss RDL—our natural-language based requirements description language—and its support for these capabilities.

3 Requirements Description Language (RDL)

Our RDL, detailed in [14], is based on the observation that the natural language used in systems' documentation already reveals semantics that can be used as a basis for both analysis of dependencies and specification of compositions that relate process specifications that span multiple documents or specify constraints that resolve inconsistencies. For this, we utilise the vast body of work on natural language grammar, semantics, and natural language processing (NLP) [28–30]. The RDL is based on scalable tool support from the WMatrix NLP engine [31], which has been shown to provide high accuracy: up to 97% for part-of-speech analysis and 93% for semantic analysis of English language texts [32].

Any document to be analysed using the RDL capabilities needs to be annotated with suitable grammatical and semantic information. This annotation is fully automated via WMatrix as the relevant information can be extracted directly from the document text. The composition specifications use these annotations as a basis of *semantic queries* which can be used both for uncovering dependencies across documents as well as specifying points of interest to specify crosscutting constraints and resolution strategies across documents. Naturally, this requires human input to encode relevant domain knowledge. The RDL is, therefore, not a substitute for a human analyst but instead a tool that can aid the complex task of studying reams of documentation and specifying constraints, resolution strategies or operational procedures that crosscut multiple documents.

The RDL was conceived as an aspect-oriented mechanism [33] to provide support for more modular representation and analysis of natural language-based requirements texts. It lends itself to analysis of SoS integration problems (and subsequent encoding of resolution strategies) as it uses:

- natural-language grammar to identify the grammatical elements that are prominent in conveying the semantics of the natural-language sentences, and, thus, are relevant for studying cross-document dependencies;
- natural-language semantics for expressing the meaning of the identified grammatical elements, and various ways of referencing them when querying the various documents under consideration;
- aspect-oriented composition mechanisms to specify crosscutting constraints and conflict resolution strategies. These constraints and strategies can be subsequently used as a basis for proof obligations or test cases for verification and validation of overall SoS behaviour.

In the following sub-sections, we first present the NLP fundamentals underpinning the RDL. We then present how this semantic and grammatical information is used as a basis for the semantic queries. Finally, we discuss how aspect-oriented composition mechanisms are utilised to specify cross-document constraints and conflict resolution strategies.

3.1 Fundamentals of the RDL

The RDL is build upon two main pillars: semantic and grammatical fundamentals. Each of these is further discussed below.

Semantics Foundations In this work, semantics refers to the meaning expressed in text. In particular, we draw on the:

- principles of similarity of meaning (i.e., synonymy) for main parts of speech groups (i.e., nouns, verbs, adjectives, and adverbs) since mostly these are the groups undertaking the main grammatical functions in a clause.
- We also use some properties of word formation (i.e., word morphology) that allow reduction of word forms.
- We propose to utilise the domain specific knowledge of entities and their dependencies captured in ontology building.
- Finally, we propose to utilise a number of semantic categories, i.e., groupings of words in accordance with their relevance to a particular classification scheme, e.g., per domain, such as words describing human activities, or Animal-related words, etc.

Each of these concepts is briefly presented below.

Synonyms are different morphological forms (i.e., words) with same sense (i.e., used with a similar or same meaning) [34, pp 70–71]. Synonymous words are generally interchangeable. For instance, in an online auction system, "to place a bid" means the same as "to make a bid", or "to bid"; or "concurrently" is the same as "in parallel" and "goods" is the same as "products". Synonymy is widely used in natural language, thus synonyms must be recognised and regarded as the carriers of the same semantics for successful natural language text analysis and understanding. For the RDL this implies that a reference to an element via its synonyms is supported.

Word Form Reduction (Lemmatisation): A given word normally has a number of possible forms, for instance "to bid", "bidding", "bids" are all about bidding and are all formed by modifications of the basic word form "bid". This most basic form of a word is called lemma. Lemmatisation (or reduction of the word to its most basic form) is widely used in natural language processing in order to simplify natural language text analysis. For the RDL, we take the view that a lemma is representative of a single part of speech only. For instance, if the text contains "bidder" as well as "to bid" we will have two lemmas, one for "to bid" as a verb and another for "bid" as a noun. Consequently, in the RDL compositions a reference to the verb's lemma will not be confused with that of the noun's [35].

From the perspective of RDL design, using lemma-based referencing allows a narrower scoping of the reference (i.e., only to all forms of the specific word), while synonym-based referencing allows for a wide scoping—to all words with a given meaning.

Ontology: an ontology is a schematisation of knowledge of a domain, representing the concepts of the domain, properties of each concept, and the relationships between the concepts. Ontologies have a number of uses, including building common understanding of information, representation, analysis, and reuse of domain knowledge, etc. However, each ontology is built to answer a specific set of questions, and, for this reason, the same domain can be represented via a number of differing ontologies [36].

For the RDL, the ontologies can be used for retrieving ontology-supported information from the requirements documents (as is done in such disciplines as information management, semantic web [37]). For instance, if the "is-a" hierarchy is represented, it could be used to identify all classes and subclasses of a given concept relevant to a given composition specification; if "part-of" relationship is provided, the ontology can help in identifying all constituents of a given stakeholder concern, etc.

Semantic Categories for Nouns and Verbs: a number of categorisations for major grammatical categories have been developed. For instance, in accordance with Quirk [34] nouns can be grouped into 5 main categories for *concrete nouns* denoting physical world entities (e.g., grass, hill, etc.), *abstract* which refer to abstract notions (e.g., happiness, friendship, etc.); *states* and *properties* reflecting mental and corporeal states and properties (e.g., hunger, pleasure, etc.); *activities* which are nouns describing activities (e.g., sale, decision, etc.). Each of these can be sub-categorised into smaller, more specialised groups.

Similarly, verbs can be categorised in accordance with their specific properties. We use one of such verb groupings for RDL, as discussed below.

Verb classifications and Role-based Interaction Patterns Several prominent results in linguistics [29, 30, 38] have shown that there is a clear link between the meaning of the words and their grammatical behaviour. Such a link can be illustrated via a simple experiment presented in [38]: 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 gallied 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 [29] suggests that the varying grammatical behaviour of verbs is the result of the differences in their meaning. Thus, using this principle, he 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 the most common use of the verb.

In our work we use the semantic categories of [29] as basis for identifying the types of relationships between concerns and, deriving composition operators.

We observe, that generally in natural language, the semantics of action-type dependencies (denoted by action-operators, or actions as per [13, 39]) are expressed by verbs or verb phrases. But, in accordance with Dixon's verb classi-

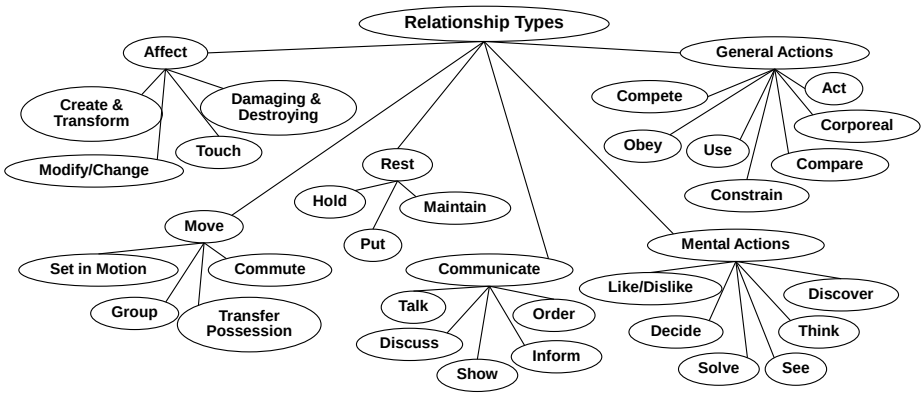

Fig. 1. Requirements analysis specific rearrangement of Dixon's verb classes

fication [29] there is only a limited set of broad meanings of verbs, thus, there must only be a limited set of broad dependency types (and, correspondingly, operators of action type) that correspond to the verb categories. There are 63 verb classes suggested by Dixon [29]. Having reviewed these categories for suitability from the perspective of composition semantics for requirements [40], we have identified the set of verb categories presented in Figure1 for use in RDL.

Grammar Foundations In natural language, a sentence is considered the highest-ranking unit of grammar [34]. Thus, we utilise a sentence and its main constituents—subject, verb, and object—to form RDL elements. One or more sentences make up a Requirement.

A *Requirement* is a description of a service the stakeholders expect of the system, the system behaviour, and constraints and standards that it should meet [41]. The requirements specified using the RDL are annotated natural language sentences. Each requirement may contain one or several clauses [34]. Each clause contains sub-elements for subject, relationship and optionally for object(s).

One or more requirement elements are encapsulated within a *concern* which is a module for encapsulating requirements pertaining to a specific matter of interest (e.g., selling and account management). A concern can be simple (containing only requirements), or composite (containing other concerns as well as requirements). Each concern is identified by its name.

A *subject* is the entity that undertakes actions described within the requirement clause. Subject in our RDL corresponds to the grammatical subject in the clause. In order to support interaction (i.e., composition) specifications involving a subject denoted with different words representing the same semantics, a set of synonymous definitions must be provided. These synonyms could be provided either through a standard synonym dictionary or per project through project specific dictionaries.

An *object* 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. Object in our RDL corresponds to the grammatical object in the clause. Usage and properties of an Object are very similar to those of the Subject. However, in a requirement there could be several objects associated with (affected by) a single subject.

Relationship depicts the action performed (state expressed) by the subject on or with regards to its object(s). Relationships can be expressed by any of the verbs or verb phrases in natural language. Using Dixon's verb categories [29], we classify relationships into a set of types (the second level nodes in Fig. 1, such as Move, etc.) and their more specific sub-types (the 3rd level nodes in Fig. 1, Set in Motion, etc.). The various relationship categories derived from Dixon's verb classification are detailed in [14, 40].

It must be noted, that we do not suggest that ALL semantics of a requirement are reduced to Subject-Relationship-Object constructs (SRO). Indeed, elsewhere we are looking at such element of requirements as degree of importance (i.e., which requirements are more urgent compared to others) or quality satisfaction, among others. However, we suggest that SROs are the main elements with respect to which the rest of the requirement semantics are formulated. Thus, SROs are the elements which participate in relations with other requirements, and are qualified, constrained or otherwise defined by both single requirement semantics, and the inter-requirements dependencies. Such semantics and dependencies can be both queried and captured in the RDL compositions, as discussed below and detailed in [14].

Tool Support for RDL The annotation of the subjects, relationships, and objects is supported via a set of links on top of the tags assigned by the Wmatrix part-of-speech (POS) tagger. The links are inserted by matching flexible patterns of POS tags. These patterns have been identified by linguists using a combination of linguistic knowledge and corpus evidence. For example, a simple rule to link a verb to its object is as follows:

```
N*o[.] (RR*/RG*/XXn3) VVN*v[.]
```

This matches the sequence 'Noun' (`N*`), followed by between 0 and 3 possibly negated 'adverbs' (`RR*/RG*/XX`), followed by a past participle 'verb' (`VVN`). In the case of a match, the noun is marked as the object of the verb. The subjects, verbs and objects are marked explicitly by Wmatrix along with the result of lemmatisation.

The annotation of verb types is realised via a mapping from the Wmatrix verb categories onto an RDL-specific tagset. In some cases the large classes of Wmatrix words were directly compatible with the RDL verb classification, for instance, the verbs of domain for Movement, Location, Travel and Transport in Wmatrix largely correspond to the RDL verbs of Motion type. On the other hand, there are semantic classes in the Wmatrix (e.g., Education, Time, etc.) tagset which have no correspondence to that of the RDL verbs tagset and their

contents have been mapped to the RDL verb classes on an individual verb by verb basis.

This annotation has been automated and is quite fast. In a recent test with a file of 56,031 words (237 pages), it took about 20 minutes to complete the initial annotation [42]. However, the quality of the annotation is significantly influenced by the quality of the input document, since every sentence of the input document is annotated with the RDL. Our POS-based patterns work rather well for well structured sentences with clearly defined subjects, verb, and objects, achieving roughly 80% of accuracy[3]. These patterns perform poorer when very long, multi-level nested sentences or grammatically incorrect sentences are used. Similarly, the quality of verb class annotation suffers if the verbs used in the text have not been covered in the RDL tagset.

3.2 Semantic Queries

The query expressions in the RDL are called semantic queries, since they select concerns/requirements on the basis of the semantics of (parts of) these concerns or requirements. The queries can use all kinds of annotations provided by the RDL, including the SRO, verb types and semantics (e.g., relationship.type= "Modify"), concern names, etc.[4] It should be noted that a requirement may have several sentences, but if one clause of one of its sentences matches the specified semantics, the requirement will be relevant for this query. Benefits of the semantic queries are twofold: firstly, we avoid syntactic matching (e.g., based on specific labels) in the queries and associated composition specifications, thus avoiding unintended element matching. Instead, queries and associated compositions are specified based on the semantics of the requirements. Secondly, it ensures that any compositions specified are semantically justified, rather than arbitrarily provided by an analyst. We provide examples of the semantic queries when discussing the composition specifications next.

3.3 Semantics-based Composition of Concerns

A composition rule in the RDL comprises three elements: *Constraint, Base* and *Outcome.* Each of these elements has an operator attribute and a semantic query expression. The query expression can select whole concerns or individual requirements from within concerns. A concern is selected if the *concern* keyword is used in the query, otherwise requirements are selected.

The *Constraint* element specifies what constraint/restriction will be placed on some requirements and what action must be taken in imposing these constraints (e.g., a conflict resolution strategy to address mismatch across con-

[3] This is an estimate based on our experiments, but is not formally validated yet. Currently we cannot provide exact time measurement for automated and manually corrected specification generations.

[4] The RDL has an XML-based syntax, which is used to automatically annotate the natural language text. For simplicity, in this paper, we omit the XML-based syntax.

stituent system specifications). The restriction that this element imposes is defined in its query expression. The action that needs to be taken is defined by the constraint operator.

An example of two Constraint elements are shown in Fig. 2 (note we omit the XML-based RDL syntax for simplification):

- Here the first Constraint element has a query *(relationship= "assign" and object = "liaison station")* stating that all requirements where a something is *assigned* to be a *liaison station* should be selected.
- The second Constraint element has a query *(subject= "liaison station" and relationship = "contact" and object= ("Assistant Section Manager" OR "Section Manager"))* stating that all requirements where liaison stations contact the Assistant Section Manager or Section Manager should be selected.

  Note, these queries do not specify where physically such requirements should be located and do not refer to any additional characteristics of these requirements. They directly point to the relevant meaning of requirements: assigning as a liaison station and the liaison station contacting the (Assistant) Section Manager.
- The "create" operator in the first Constraint specification implies that the roles of Agent, Manipulation Entity, and Target are relevant for this composition. From the Constraint query we can identify the "liaison station" as the Target; that is, something will be made into such a station.
- The "correspond" operator in the second Constraint specification implies that the roles of Speaker, Addressee, Medium and Message are relevant. From this query we also know that the Speaker is the liaison station who contacts the Addressee—here the (Assistant) Section Manager.

The *Base* element reveals the set of requirements that are affected by the elements selected in the Constraint element's query. The operator in the Base element depicts the temporal or conditional dependency between the requirements selected by the Base element query and those of the Constraint query.

An example of a Base element is shown in Fig. 2:

- The base query *(relationship="activate" and object= "RACES net")* notes that all requirements where *activation* of *RACES net* is mentioned are to be selected.
- The base operator *(meets)* denotes that immediately upon realisation of Base query, the relevant Constraint query requirement(s) must be applied. In this case, as soon as RACES net is activated, a *"liaison station"* must be assigned and the *liaison station* must *contact* the *(Assistant) Section Manger.* In addition, we can see that the RACES net will act in the Manipulation role, whereby some Actor will create a liaison station from a RACES net station as per the operator of the first Constraint.

It is worth noting that since the RDL is based on a symmetric model, it is possible to choose any set of concerns (using semantic queries) as Constraint and

any other set as Base. The same requirement may be selected by a Constraint query in one composition, and by a Base query in another. We do not discuss the automation of the actual composition process and its subsequent analysis in this paper (details are available in [14, 43] using ideas from [39]).

Finally, the *Outcome* element defines how imposition of constraint requirements upon the base set of requirements should be treated. For instance, the outcome element may specify a set of requirements that must be satisfied as post-conditions upon application of the Constraint; in this case the respective operator, such as *satisfy* will be used with that query. Unlike for Base and Constraint elements, the semantic query of the Outcome element can be empty, if no additional requirements/concerns are affected due to the Base and Constraint element interactions. In this case the *ensure* operator can be used to indicate that though there is no additional Output query, the relationships between Constraint and Base must be ensured.

4 Case Study: Reasoning about System-of-System Integration

Having discussed the three classes of SoS integration problems in Sect. 2 and presented the main elements of the RDL in Sect. 3, we now illustrate how the RDL can facilitate handling of the integration problems.

RDL works on requirements documents. Therefore, to show how RDL can be used for SoS, we would ideally possess some requirements documents where real issues of SoS integration have arisen. Unfortunately, such requirements documents are typically not available in the public domain. In the absence of requirements documents from known cases of SoS integration problems, we sampled and analysed arbitrary documents from a domain in which cases of such integration problems exist. For this paper, we selected the domain of emergency-related communications-based on the Air Florida disaster. This domain is attractive since a large number of related documents on this subject are freely available at a dedicated portal[5]. From this domain we randomly selected two: one detailing the radio communication procedures for the Virginia Emergency Net (VEN) that supports emergency communications for the state of Virginia, USA[6], and the other detailing the same for the Radio Amateur Civil Emergency Service (RACES) for the counties of Carroll, Grayson and the City of Galax in Virginia and California[7]. We treated these documents as prototypes of requirements documents as they may be used for SoS in the domain of emergency communication. We then applied the RDL to these documents analysing them for examples of managerial independence, interface incompatibility, and complexity issues. In the following, we discuss one such example for each of these integration problem classes. In the following examples, while discussing the RDL, we leave out the XML annotations for better readability.

[5] http://www.safecomprogram.gov/SAFECOM/
[6] http://www.w4ghs.org/vensop2.pdf
[7] http://www.w4ghs.org/Twin_County_SOP.pdf

It is interesting to note that for each case, our use of RDL essentially follows the same three steps:

1. Using natural-language processing, we produce a formal encoding of the two documents in RDL.
2. Using RDLs query mechanism, we search both documents for statements that may indicate potential problems arising when the two systems need to collaborate. We then check with experts and stakeholders to understand whether these statements indeed represent problems and, if so, what should be done about them.
3. We use the RDL's composition mechanism to encode any resolutions to the problems we found or to explicitly encode any conflict-resolution schemes that are already implicitly present in the documents.

4.1 Resolving Managerial Issues for SoS Integration

Since the VEN and RACES will need to cooperate to handle emergency-related communications, we need to understand who will activate the emergency procedures and how these two organisations will interact. In order to identify information related to activating these organisations, we can query the RDL-annotated document texts for the relevant information. Thus, we:

1. Find where the activation topics are treated in the documents by finding the activation related verbs, such as *activate* and its synonyms *make active, set in motion, set off, turn on, trigger, get going, trigger, prompt, initiate. Activate* is a verb of Set in Motion type which has defined roles for the *Causer* (normally taking the subject function) who sets into motion a *Moving Object* (normally taking the object function) with the optional noun phrase to specify the *Locus* (i.e., where?) role.
2. Identify what actors are filling in the appropriate roles with these action verbs:
   (a) Roles in RACES:
      i. Upon notification from the OEM or E-911 Director (Causer, though not directly defined) the *plan* (Moving Object) will be **set in motion.**
      ii. *EOC locations* (Moving Object) may be **activated** and covered with Amateur Radio (Causer) but net control should be posted outside this busy area.
      iii. An Amateur Radio Hospital Volunteer (Causer) will **activate** this *station* (Moving Object).
      iv. The Twin County RACES Emergency net (Causer) **operates** as the *Virginia/Carolina Training and Information Net* and meets every Sunday at 3:00 pm (1500 hrs) on the Fishers Peak repeater (145.130 - 600 with a tone of 103.5).
   (b) Roles in VEN:
      i. Either the SM or SEC (Causer) shall **activate** *The Virginia Emergency Nets* (Moving Object).

3. Check if there are any parallels in the activation procedure and the involved actors. In the above example:
   - The Causers are: OEM, E-911, Amateur Radio, Amateur Radio Hospital Volunteer, Twin County RACES Emergency Net, SM, SEC.
   - The Moving Objects are: plan, EOC locations, station, Virginia/Carolina Training and Information Net, Virginia Emergency Nets.
   - In points (2a: i and iv) we find that OEM, E-911 and Twin County RACES Emergency net may have some common responsibilities in activating the RACES net in an emergency. From 2b: i, we can see that the SM (Section Manager) or SEC (Section Emergency Coordinator) will activate the Virginia Emergency Net. Thus, there needs to be a protocol of interaction between these bodies to coordinate emergency handling in Virginia.
4. Since there is a need to unify the activation of these emergency bodies, there needs to be a managerial procedure to coordinate these bodies. We check if such a procedure already exists by looking at cross-references and/or collocations of the above-identified bodies in our two different documents.
   In Virginia Emergency Net document we find:
   - "VEN/D 3620 kHz (7105 & 14103.3 kHz alt) Packet, Pactor; Digital Operations - NON-RACES OPS ASM/ASEC/D"
   - "VEN/D RACES 3543 kHz (7105 & 14103.3 kHz alt) Packet, Pactor, Digital Operations - ONLY REAL RACES OPS ASM/ASEC/D"

   Here we also find that ASM/ASEC/D corresponds to Assistant Section Manager/Assistant Section Emergency Coordinator for Digital Operations. Thus, from this document we have identified that the Assistant Section Manager is assigned to the communications involving RACES operations.
   In RACES document we find:
   - "Liaison stations to the following National Traffic System nets will be assigned (Old Dominion Emergency Net-3947) (Old Dominion Emergency Net 7240) (***Virginia Emergency Net*** 3910)."

   Here we have identified that a liaison station will be assigned to communicate with the Virginia Emergency Net.
   Thus a procedure of communication between these two systems has become clearer: it transpires that the Assistant Section Manager or Section Manager will be responsible for managing the communication between VEN and RACES via a RACES liaison station.
5. Finally, we assert this communication procedure by defining a specific composition (Fig. 2).

Thus, by using the synonym-based querying of the RDL we were able to identify the areas in the input documents where issues related to activation were discussed. We then identified entities that carry out same roles for the activation process in different documents and were able to consider their relations to each other in the management of the two systems and their co-working. Based on these considerations, we then defined a composition to assert a managerial process for the interaction of the two systems.

```
Composition: VEN_RACES_Communication
Constraint: create
    operator: enable
    query: (relationship= "assign" and object = "liaison station")
Constraint:
    operator: correspond
    query: (subject= "liaison station" and relationship = "contact" and
           object= (Assistant Section Manager" OR "Section Manager"))
Base:
    operator: meets
    query: (relationship="activate" and object= "RACES net")
Outcome:
    operator: ensure
```

Fig. 2. Composition for Communication Procedure between VEN and RACES

4.2 Addressing Incompatible Interfaces in SoS Integration

The second problem in SoS integration, discussed in Sect. 2.2 of this paper, relates to the incompatibility of interfaces between systems. We will now consider how such incompatibility requirements can be identified and resolved with our RDL-based approach.

The previously discussed integration failure in the case of Air Florida Flight 90 was caused due to the use of different communication frequencies by different emergency teams. Let us now check if such a problem may arise in integration of the VEN and RACES systems. To do this, we need to:

1. Verify that the frequencies listed against each of the nets are consistent across documents. Here we use domain knowledge about the format of describing nets and their frequencies by listing the frequency immediately before/after the name of the net—for example, "Old Dominion Emergency Net 7240"—without explicitly using the term "frequency" or its synonyms in the description.
2. Look up where use of frequencies is explicitly mentioned by looking up the term *frequency* and its synonyms, such as *Hz* and checking with stakeholders/managers that these are correct for each net.

In these two example documents we have a number of references to nets and tier respective frequencies, such as:

- In the RACES document there are references to:
  - Old Dominion Emergency Net 7240
  - Virginia Emergency Net 3910
- In Virginia Emergency Net document there are references to:
  - 3543 kHz (7105 & 14103.3 kHz alt) RACES
  - 7240 kHz Alt . . . ODEN (i.e., Old Dominion Emergency Net)

Thus the reference from the RACES document to ODEN frequencies is consistent with that from the Virginia Emergency Net.

The next reference from RACES document is to Virginia Emergency Net ***3910.*** However, in the document on Virginia Emergency Net there is no mention of a frequency of 3910. Instead, it refers to ***3543*** kHz (***7105*** & ***14103.3*** kHz alt) frequencies for RACES. This obviously is an area that needs clarification with the stakeholders:

- do the documents have the correct *different* frequencies listed, or
- is there an incompatibility in the specification and these documents must use the *same* frequency? If there is an incompatibility here, what is the correct frequency to be used?

For this example we assume that the communication protocols used require that the stations use the same frequency to communicate. In which case, we have identified an integration problem for which a resolution decision must be made. Let us assume that the specification in the document for the Virginia Emergency Net is chosen as the correct one. We can now define a composition specification, as shown in Fig. 3, asserting that the values of frequencies for VEN listed in the VEN document (frequency.value=doc.VEN and object="VEN") will dominate over those listed in the RACES document (frequency.value=(doc.RACES and object= "VEN")). This composition will ensure harmonised resolution of this issue in future requirements.

```
Composition: VEN_RACES_Frequency_Harmonisation
Constraint:
    operator: modify
    query: frequency.value=doc.VEN and object="VEN"
Base:
    Operator: concurrent
    Query: frequency.value=(doc.RACES and object= "VEN")
Outcome: ensure
```

Fig. 3. Composition for Interface Harmonisation between VEN and RACES

4.3 Support for Handling Complexity in Systems' Documentation

We have previously discussed the issue of complexity in understanding behaviours of the systems to be integrated. From the perspective of requirements analysis, this complexity manifests in the need to identify and understand the behaviours of interest from a large volume of written documentation (since our discussion relates to the documents written in natural language). Thus, here the complexity mainly manifests itself in the need to treat large volumes of information.

The utility of our approach lies in the ability to:

1. *identify and separate concerns of interest* from the rest of the documentation and
2. *define localised compositions* to assert a particular set of rules/interaction resolutions that crosscut multiple documents and/or concerns within these documents.

This aspect-oriented basis of the RDL provides inherent support for modular (studying a particular concern of interest in isolation from other concerns in a system, in this case a SoS) and compositional reasoning (reasoning about the interdependencies and relationships amongst concerns to understand the emergent behaviour, in this case the behaviour of the SoS) [44].

An example of such modular and compositional reasoning was illustrated in Sect. 4.1, whereby a particular behaviour related to activation was studied independently from the rest of the documents' contents. This behaviour was considered in tandem with the set of entities required to carry out the given behaviour in different documents and the relationships of these entities were also considered. In Sect. 4.2, we also presented an example composition definition which was used to harmonise differences between the two input documents.

The example documents in this case study are largely unstructured natural language texts. The RDL compositions can also be used to define crosscutting behaviour across sets of more structured requirements artifacts, such as use case specifications. For instance, a behaviour related to sending message will be discussed in a number of use cases detailing the RACES and VEN systems. A single localised composition may be used to specify that an encryption protocol should be employed at any point when a message is sent. Such a composition specification will not require any change/rewriting of the existing use cases. Yet, when the relevant use cases are viewed/analysed in an appropriately tooled environment, the additional detail on encryption use will be incorporated across the relevant use case steps. Further details on the crosscutting nature of the RDL compositions is provided in [14, 45].

5 Discussion

In Sect. 4, we have discussed how our RDL can be used to study the three major classes of integration failures in SoS: managerial independence, incompatible interfaces and complexity of constituent systems. Significant benefit can also be derived from our RDL-based approach when we consider a set of documents containing natural language requirements for which a large number of economic, social, political factors of heterogeneity must be observed, constraints enforced and priorities maintained. In these cases the semantics-based queries of the RDL can assist in direct identification of relevant points in the documents where, for instance, a particular policy is discussed, etc. Moreover, when needed, a locally defined composition specification can be used, for instance, to enforce a new policy, or introduce the behaviour of new system into a broad set of existing operations specification documents.

However, there are a number of other SoS-characteristic problems that can occur during the analysis and cross-examination of specification documents. Below we discuss some of them as well as some potential extensions of our RDL approach to address them:

1. *Different languages:* Because of the geographic distribution of collaborating systems in a SoS, their specification documents may be written in different languages e.g. one set of specification documents may be in English and the other in German.
   To handle such differences the RDL approach can be furnished with a cross-language analysis support. Such a cross-language support is indeed feasible, since all the grammatical and semantic features of the RDL are manifest in the vast majority (if not all) human languages whereby an entity (subject) carries out some activity (verb) on or in respect with other entities (objects). Moreover, the verb classes identified by Dixon, and used in the RDL, are also largely language-independent, as discussed in [29].
2. *Regional syntax:* It is a well known fact that the same language may have regional dialects. These dialects can become so well rooted as to make their way in to the written word and thus establish themselves as different versions of the same language. A well-known example is the difference between the US and UK versions of the English language. For example two different words—soccer and football—refer to the same sport in the US and UK respectively. However, the same word football refers to two different sports in the US and UK.
   The RDL approach can be furnished with support to identify the dialects and automatically establish correspondences between relevant entities. For this we will need to build a language corpus for each relevant dialect and provide a set of algorithms which would identify the dialect used in each document from the natural language clues. Clues such as use of words/expressions unique to a given dialect (e.g., *"fall semester"* in US English); spelling and/or grammar peculiarities (e.g., "behavior vs. behaviour"), frequencies of particular word use, etc. can be utilised for the purpose.
3. *Domain specific syntax:* This problem can be further divided into two subcategories.
   (a) Domain specific jargon in specifications of the collaborating systems: certain words like "Tympanum" from the "Anatomy" domain may not be understandable to non-experts.
   (b) Different domain specific meaning of the same word: for example, the word "Delta" means different things in Geology and Mathematics.

   The first of these problems can be resolved by providing domain specific lexicons and/or ontologies, where required. However, building these can be a substantial effort in itself. This can be facilitated (or substituted) with a set of algorithms which could use the previously discussed techniques (i.e., most frequently used concepts, unique words, and combinations etc.) to identify the general domain of the document and to provide a summation of relevant term occurrences and/or definition from a set of sample documents of that domain (e.g., by obtaining these from the Web).

The resolution of the second of these problem can already be supported via word sense disambiguation techniques used in NLP, such as realised in the Wmatrix [31] tool.

A number of other problems, such as *varying formats of specification* documents (e.g., some written using use case specifications, some using viewpoint-based templates); *inconsistent detail* (e.g., some systems may be specified in great detail while others may have almost no documentation); *misinterpretation of data* (e.g., in one specification a mean value is used as representative, while in the other system a mode), etc. will also arise in distributed heterogeneous SoS. While these require further research and development effort, it could be relevant to note that the RDL would be well positioned in supporting such problem resolutions since:

- The RDL is based on natural language characteristics and does not require any other specific format or restrictions;
- It has been shown that the RDL approach is amenable to automation [14], and already has automated support for a number of processing activities [14, 43];
- The RDL uses the semantics of natural language to identify relevant portions in the different system specification documents, allowing domain experts to focus only on details relevant to their work;
- The RDL compositions are able to support localised specifications of cross-document dependencies/constraints.

6 Related Work

Ultra large scale (ULS) systems have been defined in [2] as "A system at least one of whose dimensions is of such a large scale that constructing the system using development processes and techniques prevailing at the start of the 21st century is problematic." These futuristic systems will exhibit characteristics not unlike SoS today. These characteristics include decentralized control, conflicting requirements, heterogeneity, evolution, failures of parts of the system, and erosion of the people/system boundary etc. [2] Problems similar to the ones facing the requirements analysis of SoS have also been raised for ULS systems. These include finding compatibility, redundancy, inconsistency, emergent properties, in requirements and reasoning about requirements in the presence of uncertainty and ambiguity [2]. Therefore, the approach presented in this paper, may also be relevant for ULS systems.

A number of approaches have been proposed in recent years focusing on the use natural language processing and information retrieval techniques for analysis of crosscutting concerns. The EA-Miner tool [45, 46] uses the WMatrix toolset to mine for crosscutting concerns in natural language requirements specifications. EA-Miner is a tool that provides integrated support for the RDL, generating RDL specifications from the mined (and, subsequently edited) requirements model.

Other relevant works include Theme/Doc LSA [47] and Repertory Grid [48]. Theme/Doc LSA uses the Latent Semantic Analysis technique [49] to build a concern-requirement matrix. In the matrix concerns correspond to terms while requirements correspond to documents used in the analysis. The LSA algorithm is then used to identify relevant concern-requirements associations. Threshold values can be set to filter out associations that are less pertinent or not of interest. However, too low a threshold value can lead to cluttered concern-requirement association graphs while too high a value can lead to more sparsely populated graphs. We have found that a hybrid approach, combining LSA with NLP, yields better results as the NLP-based analysis can be used to identify elements of interest which can then be subject to an LSA analysis for identifying relationships [50].

The work of Niu and Easterbrook [48] is based on the Repertory Grid technique from Psychology which aims at capturing how people construct mental models of objects in their environment. Using this technique, one can study how two constructs may be similar or different in a particular document set, hence identifying the contributions of specific tasks to high-level system goals. However, in contrast to our approach, this technique requires structured requirements as input and the construction of the grid is not automated. However, its integration with the RDL can yield fruitful results by allowing one to study the mental model of a SoS (for instance, based on observations from ethnographic studies) from the perspective of the various stakeholders of the constituent systems. This can facilitate a more top-down analysis of SoS integration challenges compared to the bottom-up documentation-based analysis supported by our approach.

7 Conclusion and Future Work

The conception and development of a SoS poses a number of challenges, not least due to the fact that SoS are created opportunistically owing to some social or business need. Since the constituent systems are often not designed to work together in the first place or, when they are they remain under the control of different autonomous organisations, incompatibilities are almost inevitable owing to the inherent complexity of the individual systems, their managerial independence, and past (often divergent) evolution trajectories. Reasoning about the overall behaviour of a SoS is, therefore, non-trivial in the presence of such diversity and heterogeneity. We cannot escape the fact that mostly such reasoning is based on reading reams of documentation about the individual systems-almost 80% of system specifications remain in natural language. The simple search facilities in word processing or file rendering systems are not able to relate concepts that may have been referred to using different terms. Furthermore, it is not possible, without specialist tools that create vendor lock-ins and require use of specific specification notations, to specify constraints across the documentation.

Our RDL presents a solution to these challenges. It is based on well-established natural language processing concepts and can be deployed across domains. The semantic queries in the RDL work on rich information clues already inherent

in natural language specifications hence making it possible to relate concepts as well as specify composition rules that work on this natural language semantic basis. This, in turn, means that the composition rules are resilient to changes in documentation structure. Most significantly, the RDL approach can be utilised with any documentation written in natural language. Our case study has shown that the approach can uncover integration issues in real-world SoS and can be used to specify suitable resolution strategies.

We do not consider the RDL to be the final solution to these challenges. Instead we see it as a stepping-stone towards scalable mechanisms for reasoning about SoS integration issues. In Sect. 5 we have identified a number of future work paths for this work. They will be our focus in the long-term. In the short-term, we aim to apply the RDL to further case studies of real-world SoS to gather more empirical data about its efficacy based on a larger corpus of SoS examples.

We are also currently working to develop a more detailed, hierarchical taxonomy of expected SoS integration problems and faults. Such a taxonomy may help to focus the attention of the SoS engineer towards finer grained problems. These problems include, but are not limited to, technical incompatibilities, quality of service issues, and regulatory compliance problems. Once developed, this taxonomy can be used as the basis of a library of RDL query templates. These templates can assist the SoS engineer by lessening the work of writing queries for expected SoS problems. The SoS engineer will of course have to customise or instantiate the templates for the particular problem at hand. For example, a query template can be written to help the SoS engineer query the system's documentation to find out what kind of communication protocols are used or supported by the system. The template method is expected to be useful in performing quick, cursory comparisons of different requirement and specification documents. Detailed analysis of requirement documents is still expected to require writing unique, standalone queries in RDL.

Acknowledgements

This work is supported by EC FP6 project, AMPLE: Aspect-Oriented Model-Driven Product Line Engineering and the EC FP7 project DiVA: Dynamic Variability in Adaptive Systems. Awais Rashid is also supported by a Chair Regionale by the Pays de la Loire Regional Government in France.

References

1. Sage, A.P., Cuppan, C.D.: On the systems engineering and management of systems of systems and federations of systems. Information, Knowledge, Systems Management **2**(1) (2001) 325–345
2. Northrop, L., Feiler, P., Gabriel, R.P., Goodenough, J., Linger, R., Longstaff, T., Kazman, R., Klein, M., Schmidt, D., Sullivan, K., Wallnau, K.: Ultra-Large-Scale Systems: The Software Challenge of the Future. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA (July 2006)

3. Boardman, J., Sauser, B.: System of systems: The meaning of of. In: IEEE Int'l System of Systems Conf. (April 2006) 118–123
4. DeLaurentis, D., Callaway, R.: A system-of-systems perspective for public policy decisions. Review of Policy Research **21**(6) (2004) 829–837
5. DeLaurentis, D.: Role of humans in complexity of a system-of-systems. In Duffy, V.G., ed.: Digital Human Modeling, HCII 2007. Volume 4561 of LNCS. (2007) 363–371
6. Jamshidi, M.: System of Systems Engineering: Innovations for the 21st Century. John Wiley & Sons, Inc. (November 2008)
7. Keating, C., Rogers, R., Unal, R., Dryer, D., Sousa-Poza, A., Safford, R., Peterson, W., Rabadi, G.: System of systems engineering. EMJ – Engineering Management Journal **15** (2003) 36
8. Sage, A.P.: Conflict and risk management in complex system of systems issues. In: IEEE Int'l Conf. on Systems, Man and Cybernetics. (2003)
9. Eisner, H.: RCASSE: rapid computer-aided systems of systems engineering. In: 3rd Int'l Symposium of the National Council of System Engineering (NCOSE). (1993) 267–273
10. Kotov, V.: Systems of systems as communicating structures. Technical report, Hewlett Packard Computer Systems Laboratory Paper HPL-97-124 (1997)
11. Popper, S.W., Bankes, S.C., Callaway, R., De-Laurentis, D.: System of systems symposium: Report on a summer conversation. In: 1st System of Systems Symposium. (2004) Available from http://www.potomacinstitute.org/academiccen/SoS Summer Conversation report.pdf.
12. Baniassad, E.L.A., Clements, P., Araujo, J., Moreira, A., Rashid, A., Tekinerdogan, B.: Discovering early aspects. IEEE Software **23**(1) (2006) 61–69
13. Rashid, A., Moreira, A., Araujo, J.: Modularisation and composition of aspectual requirements. In: International Conference on Aspect-Oriented Software Development (AOSD), ACM (2003) 11–20
14. Chitchyan, R., Rashid, A., Rayson, P., Waters, R.W.: Semantics-based composition for aspect-oriented requirements engineering. In: International Conference on Aspect-Oriented Software Development (AOSD), ACM (2007) 36–48
15. Maier, M.: Architecting principles for systems of systems. Systems Engineering **1**(4) (1998) 267–84
16. Madni, A.M., Sage, A.P., Madni, C.: Infusion of cognitive engineering into systems engineering processes and practices. In: IEEE Int'l Conf. on Systems, Man and Cybernetics. (October 2005) 960–965
17. House of Commons Transport Committee: The opening of Heathrow Terminal 5. Twelfth Report of Session 2007-08. HC 543, Published on 3 November 2008 by authority of the House of Commons London. Downloaded from: http://www.publications.parliament.uk/pa/cm200708/cmselect/cmtran/543/543.pdf on Dec. 16, 2008 (2008)
18. BBC News: What went wrong at heathrow's T5? Downloaded from: http://news.bbc.co.uk/1/hi/uk/7322453.stm on Dec. 16, 2008 (March 2008)
19. Thomson, R.: British airways reveals what went wrong with Terminal 5. Downloaded from: http://www.computerweekly.com/Articles/2008/05/14/230680/british+airways+reveals+what+went+wrong+with+terminal.htm on Dec. 16, 2008 (May 2008)
20. Ellison, R.J., Goodenough, J., Weinstock, C., Woody, C.: Survivability assurance for system of systems. Technical report, CMU Software Engineering Institute (May 2008) CMU/SEI-2008-TR-008, ESC-TR-2008-008.

21. Rechtin, E.: Systems Architecting. Prentice Hall, Upper Saddle River, NJ (1990)
22. Navarro, L.D.B., Südholt, M., Douence, R., Menaud, J.M.: Invasive patterns for distributed programs. In: Proc. 9th Int'l Symposium on Distributed Objects, Middleware, and Applications (DOA'07), Springer (November 2007) 772–789
23. Smith, B., Tolman, T.: Can we talk? Public safety and the interoperability challenge. National Institute of Justice Journal (April 2000) 17–21
24. The Office for Interoperability and Compatibility (OIC),The Department of Homeland Security: The system of systems approach for interoperable communications. Obtained from http://www.safecomprogram.gov/NR/rdonlyres/FD22B528-18B7-4CB1-AF49-F9626C608290/0/SOSApproachforInteroperableCommunications_02.pdf, last visited January 2008 (2008)
25. Mars Climate Orbiter Mishap Investigation Board: Phase I report. ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO_report.pdf (November 1999)
26. NASA: Mars climate orbiter official website. http://mars.jpl.nasa.gov/msp98/news/mco990930.html (1998)
27. Marshall, S.: Software engineering: Mars climate orbiter. http://www.vuw.ac.nz/staff/stephen_marshall/SE/Failures/SE_MCO.html
28. UCREL: UCREL semantic analysis system (USAS). http://www.comp.lancs.ac.uk/ucrel/usas/: Lancaster University, UK (2006)
29. Dixon, R.M.W.: A Semantic Approach to English Grammar. 2nd edn. Oxford University Press (2005)
30. Levin, B.: English verb classes and alternations: a preliminary investigation. The University of Chicago Press (1993)
31. Rayson, P.: WMATRIX. Lancaster University, URL: http://www.comp.lancs.ac.uk/ucrel/wmatrix/ (2007)
32. Sawyer, P., Rayson, P., Cosh, K.: Shallow knowledge as an aid to deep understanding in early phase requirements engineering. IEEE Trans. Software Eng. **31**(11) (2005) 969–981
33. Filman, R.E., Elrad, T., Clarke, S., Akşit, M.: Aspect-Oriented Software Development. Addison-Wesley Professional (2004)
34. Quirk, R., et al.: A Comprehensive Grammar of the English Language. Longman; London; New York (1985)
35. Francis, W.N., Kučera, H.: Frequency Analysis of English Usage: Lexicon and Grammar. Houghton Mifflin, Boston (1982)
36. Noy, N.F., McGuinness, D.L.: Ontology development 101: A guide to creating your first ontology. Technical report, Stanford Knowledge Systems Laboratory and Stanford Medical Informatics (2001) Technical Report KSL-01-05 and SMI-2001-0880.
37. Fensel, D., Hendler, J.A., Lieberman, H., Wahlster, W.: Spinning the Semantic Web. The MIT Press (2002)
38. Hale, K.L., Keyser, S.J.: A View from the Middle. MIT, Center for Cognitive Science, Cambridge, MA (1987)
39. Moreira, A., Araujo, J., Rashid, A.: Multi-dimensional separation of concerns in requirements engineering. In: 13th IEEE Int'l Conf. on Requirements Engineering (RE 05). (2005) 285–296
40. Chitchyan, R., Rashid, A.: Tracing requirements interdependency semantics. In: Workshop on Early Aspects (held with ASOD 06), Bonn, Germany. (2006)
41. Sommerville, I.: Software Engineering. 2nd edn. Addison-Wesley (2004)

42. Sampaio, A., Rashid, A., Chitchyan, R., Rayson, P.: EA-Miner: Towards automation in aspect-oriented requirements engineering. Transactions on Aspect-Oriented Software Development **3** (2007) 4–39
43. Waters, R.W.: MRAT – the multidimensional requirements analysis tool. Master's thesis, Computing Department, Lancaster University (October 2006)
44. Rashid, A., Moreira, A.: Domain models are NOT aspect free. In: Model Driven Engineering Languages and Systems. Volume 4199 of LNCS., Springer (2006) 155–169
45. Chitchyan, R., Pinto, M., Rashid, A., Fuentes, L.: COMPASS: composition-centric mapping of aspectual requirements to architecture. Transactions on Aspect-Oriented Software Development **4** (2007) 3–53
46. Sampaio, A., Chitchyan, R., Rashid, A., Rayson, P.: EA-Miner: a tool for automating aspect-oriented requirements identification. In: 20th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE'05), New York, NY, USA, ACM (2005) 352–355
47. Kit, L.K., Man, C.K., Baniassad, E.: Isolating and relating concerns in requirements using latent semantic analysis. SIGPLAN Not. **41**(10) (2006) 383–396
48. Niu, N., Easterbrook, S.M.: Analysis of early aspects in requirements goal models: A concept-driven approach. Transactions on Aspect-Oriented Software Development **3** (2007) 40–72
49. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley (1999)
50. Alves, V., Schwanninger, C., Barbosa, L., Rashid, A., Sawyer, P., Rayson, P., Pohl, C., Rummler, A.: An exploratory study of information retrieval techniques in domain analysis. In: 12th Int'l Software Product Line Conf. (SPLC'08), Washington, DC, USA, IEEE Computer Society (2008) 67–76