Overlaps among Requirements Specifications

George Spanoudakis, Anthony Finkelstein

Department of Computer Science,

City University

Northampton Square, London EC1V 0HB, UK

email: {gespan | acwf @cs.city.ac.uk}

Abstract. Although overlap between specifications – that is the incorporation of elements which designate common features of the domain of discourse – is a prerequisite for specification inconsistency, it has only been a side concern in requirements engineering research. This paper discusses overlap, points out the complicating factors in its identification an outlines the key components of a programme of research in this area.

1. Motivation

In software engineering settings where different stakeholders may have requirements about the system to be built, there is no point for worrying about inconsistency among specifications of these requirements unless you are pretty confident that they refer to the same things within the shared domain of discourse. In other words, unless there is an overlap between them. This paper is concerned with such overlaps, discusses the problems encountered in their identification and proposes an agenda for a research programme aiming at effective solutions to these problems.

Specifications overlap if one or more of their components designate common aspects of the system under development, its domain and environment, and the process of construction. The existence of an overlap may imply *consistency rules* between specifications, which if breached make them inconsistent. A breach of a consistency rule generally manifests itself as a logical inconsistency, that is the assertion of a fact and its negation for a pair of overlapping components. Overlap and inconsistency are two different levels of interference between specifications, the first of which is prerequisite for the second (Finkelstein et al [1996]).

Consider for example the specifications in Figure 1. They describe loans from some university library, they have been expressed in different languages (*Specification-1* in an ER-modelling notation and *Specification-*

2 in plain English), and they have been constructed by different agents (Anthony and George). The *Staff* component in *Specification-1* overlaps with the *Student* component in *Specification-2*, if there are members of the university staff who are also students. Given this overlap between *Student* and *Staff*, the specifications in Figure 1 would be inconsistent with respect to, let's say, a consistency rule requiring that there should be a single defined amount of money that any library borrower should pay for late returns to the library (they give different late return penalties for students and staff members). Note that, it would be meaningless to check this consistency rule if the specifications in Figure 1 were describing loans from libraries of different universities and therefore there was no overlap between them. Also, checking the same consistency rule would reveal no inconsistency if the specifications were concerned with the same university library but no student could be a member of the staff at that university. In this case *Student* and *Staff* would not overlap.

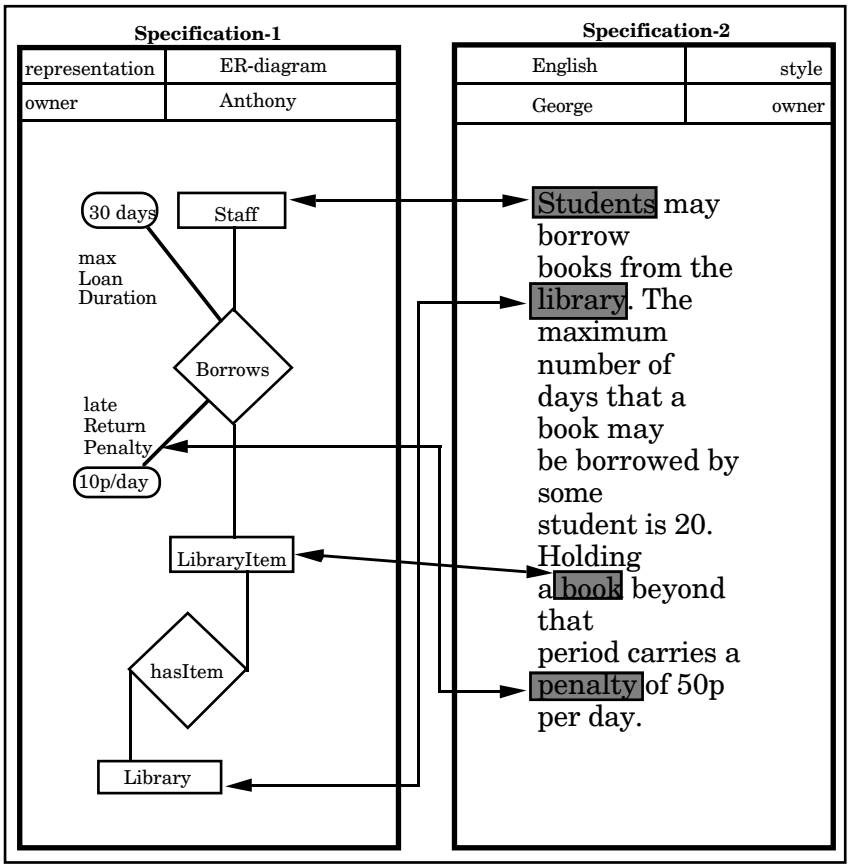

Figure 1: Interfering specifications

Evidently from this example, overlaps have to be detected and made explicit before checking for inconsistencies, and reporting on the consistency status of specifications has to be made in reference to specific overlap relations.

2. What is overlap ?

Overlap may be formally defined as a relation between the interpretations of the components of two specifications (Spanoudakis et. al. [1997]), and different types of overlap relations may be distinguished. In particular, a pair of specification components:

– *overlap totally* if the sets of the objects they designate are the same;
– *overlap partially* if their designated sets have both elements in common and non common elements;
– *overlap inclusively* if one of the designated sets is a proper subset of the other; and
– *do not overlap* at all if their designated sets have no elements in common.

The distinction among different types of overlap relations is important not only because these may have a different impact on the consistency status of two specifications but also because the resolution of inconsistencies occurring due to each of these types of overlap might not be the same.

Let us consider again the example of Figure 1 and assume that the properties of a superordinate specification component may be overridden by the properties of the components it subsumes. If *LibraryItem* in *Specification-1* overlaps totally with *Book* in *Specification-2* then the two specifications are inconsistent with respect to the rule about the single amount of money that should be paid for late returns to the library. However, if *LibraryItem* overlaps inclusively with *Book* (that is the set of objects designated by *Book* is a subset of the set of objects designated by *LibraryItem*) then the rule is not violated because of the overriding of the penalty by the subordinate component (*Book*). In this case 10p will be the penalty to be paid for library items in general and 50p will be the penalty to be paid for books in particular.

Similarly the way to resolve inconsistencies depends on the type of overlap. If the overlap between *Book* and *LibraryItem* is total then only one of the specified penalties should be retained or alternatively a new single common penalty should replace both of them. On the other hand if *LibraryItem* overlaps inclusively with *Book* and no overriding of properties is allowed, a third possibility would be to resolve the inconsistency by allowing overriding.

The above four types of overlap are exhaustive in covering all the possible relations between the interpretations of two components.

3. Overlaps and agents

Note that in reality the interpretations of specifications cannot be enumerated with sufficient completeness nor be defined in sufficient precision. Although it is always possible to express axioms restricting considerably these interpretations (and in some cases rule out the possibility of a common interpretation) rarely these axioms are susceptible to just one interpretation. As a consequence it is impossible to decide about interpretation relationships and subsequently component overlap solely on the basis of such axioms.

To cope with this problem, we suggest that overlap relations need to be understood in terms of interpretations *intended* – as opposed to enumerated or defined – for specifications by specific agents. The assertion of an overlap relation between two specification components by an agent is essentially a confirmation that the interpretations the agent ascribes to these components are related as indicated by the type of the overlap relation. However the agent does not have to define or enumerate the interpretations which give rise to these relations.

Our notion of agent, as the source of interpretations underlying overlap relations, includes humans (specification owners, third parties) and schemes of deducing overlap relations. Agents may assert overlap relations jointly. In this case, the interpretations underlying overlap relations are the interpretations that the agents jointly ascribe to the specifications. Since different agents may assert different overlap relations between specifications, inconsistency has to be checked in respect to overlap relations asserted by specific agents and the consistency status of specifications may differ across overlaps asserted by different agents. We give a formal account of this notion of inconsistency elsewhere (Spanoudakis et. al. [1997]).

4. Identification of overlap

Despite the importance of overlap for inconsistency, its identification has only been a side concern in relevant requirements engineering research. In reported work overlaps are usually identified by representation conventions, shared ontologies and specification owners.

The simplest and most common representation convention is to assume total overlaps between components with identical names and no overlaps between any other pair of components (Easterbrook *et. al.* [1994], Finkelstein *et. al.* [1994], Easterbrook and Nuseibeh [1996], Heitmeyer *et. al.* [1995], van Lamsweerde

[1996], Nissen *et. al*. [1996]). No doubt this convention is very weak regarding even simple forms of specification heterogeneity such as the presence of synonyms and homonyms. Overlaps may also be identified by relations *embedded* in the representation language of specifications. For instance, the "Is-a" relation in various object-oriented specification languages, which is usually given a *set-inclusion semantics*, is a statement of overlap: the subtype designates a subset of the instances of the supertype. Similarly, the implication ( ) between two predicates of the same arity constitute a statement of inclusive overlap in predicate logic. Note that such embedded relations would anyway need (like overlap relations) to be identified explicitly for independently constructed specifications. The problem with them is that they may only interrelate specifications represented in the same language.

The second approach uses shared ontologies for assigning interpretations to specifications or specification components and identifies overlaps between them only if they have been "tagged" with the same item in the ontology (Leite & Freeman [1991], Robinson & Fickas [1994], Boehm & In [1996]). There is an inherent problem with this approach. In order to be useful – that is to facilitate communication between agents about a domain of discourse without necessarily assuming a global shared theory – ontologies have to include definitions of their items and be general; and agents need to "commit" themselves to the ontology in the sense that their observable actions are consistent with the definitions of the items in it (Grueger [1993], Guarino [1994]). As a consequence of ontology generality, specifications need to add a lot more details to it for describing a system and its domain in reasonable completeness. Inevitably this leads to associations of many specification components with the same item in the ontology and as a result only coarse-grain overlaps could be identified using these associations. Also if ontologies incorporate item definitions then they are domain models, and therefore it is likely to be interpreted in different ways by different specification owners. As a consequence the same ontology item may be used for ascribing different meanings to different specification components. In such cases it is not safe to assume the existence of an overlap between components associated with the same item in the ontology unless there is evidence that the specification owners understand the ontology in the same way.

A third approach is to involve humans in overlap identification (Easterbrook [1991], Zave & Jackson [1993], Spanoudakis & Finkelstein [1997]). In this approach, specification owners (or a third party) explore their specifications and identify overlaps possibly with the support of some tool or method. This support might range from visual aid for the graphical selection and recording of overlaps (Easterbrook [1991]) to specification matching methods (Spanoudakis & Finkelstein [1997]). Simple visual aid to, otherwise, manual comparisons is not sufficient when it comes to specifications of substantial complexity and matching methods tend to be sensitive to heterogeneity in the specification representation, granularity and level of abstraction (Spanoudakis & Constantopoulos [1995]).

Note also that, all these three approaches even when they manage to distinguish between components which overlap and components which do not, they fail to distinguish among the different types of overlap.

5. Research Programme Components

As a prerequisite of specification inconsistency overlap has to be a central concern in a research programme whose objective is the management of interference in requirements engineering. The need to deal effectively with overlap makes necessary the incorporation of some key components to such a research programme, including:

Overlap identification support – It should be clearly appreciated that the identification of overlap is complicated by specifications, which might have been constructed independently, by stakeholders with varying concerns, backgrounds and knowledge. As a consequence they might have been expressed in different languages; might be at different levels of abstraction, granularity and formality; and might deploy different terminologies. The complexities arising from these forms of heterogeneity between specifications – set alongside the normal software engineering problems of scale – make the identification of overlap a very complex activity. Identifying overlaps will always require human intervention. However humans would undoubtedly benefit from automated reasoning, tool and method support particularly in settings characterised by problems of scale. To advance the current state of the art, any tool or method whose objective is to provide automated support to this identification should deal effectively with the problems introduced by specification heterogeneity regarding granularity and levels of abstraction.

Synthesis of overlap relations: In some settings a combination of overlap identification methods might be required, and overlap relations that have been identified by different methods or agents, possibly at different times, might need to be synthesised. For instance, in settings where a shared ontology has been evidently used coherently in the past, it may make sense to use it for identifying coarse-grain overlaps and then refine them using some other method (e.g. specification matching). Or, in checking the consistency of more than two specifications it might be necessary to rely on pairwise overlaps between them asserted by different agents. In such cases, we need to be able to check if the overlap relations asserted by different agents are consistent. It is also necessary to have a coherent and systematic way of synthesising overlap relations in order to inform consistency checking.

Overlap representation and management: Overlap relations need to be "tagged" by the agents that asserted them; and represented separately from the specifications they concern, in forms suitable for consistency checking. They also need to be maintained after consistency checking. Representing and maintaining them in this way makes the checking of consistency with respect to different opinions on overlap easier and

facilitates the role of overlap as an indicator of the potential for inconsistency when changes to specification components violate previously satisfied rules or additional consistency rules are identified. However, this sort of representation introduces interesting questions about what happens to asserted overlap relations when specifications change, especially in cases where the owners of specifications are different from the agents that asserted the overlap relations.

Formal foundations: A formal account of overlap is required as a basis for identifying, representing, maintaining and synthesising overlaps. Crucial questions like, the consistency of overlap relations (asserted by the same or different agents), the ability to infer further overlap relations from asserted ones, and the reliability of consistency checking on the basis of overlap relations that have been asserted by different agents, might only be answered after some grounding work on the formalisation of overlap and its relationship to agents.

Our current work focuses on issues falling into all these four components. We are improving our "reconciliation" method (Spanoudakis & Finkelstein [1997]), which supports the co-operative identification of overlap between specifications expressed in a particular class of object-oriented languages, in the light of a case study based on the LAS inquiry report [1993]. We are also investigating general characteristics of processes for guiding specification owners in overlap identification in distributed settings, including the time and the mode of their application. And finally, we are working towards a formal foundation of overlap relations (Spanoudakis et el. [1997]).

6. Conclusions

This position paper has argued about the need to identify overlaps before dealing with inconsistency among requirements specifications and has suggested that overlaps must be understood in terms of relationships between interpretations intended for specifications by specific agents. It has also discussed the difficulties in identifying overlaps and briefly reviewed related work. Finally, it has sketched the components that should be incorporated in a research programme, whose objective is the management of interference in requirements engineering, if this programme is to deal effectively with the problem of overlap identification.

References

Boehm B., In H., 1996. "Identifying Quality Requirements Conflicts", IEEE Software, March 1996, pp. 25-35.

Easterbrook S., 1991. "Handling Conflict between Domain Descriptions with Computer-Supported Negotiation", Knowledge Acquisition, 3, pp. 255-289.

Easterbrook S., Finkelstein A., Kramer J. & Nuseibeh B., 1994. "Co-Ordinating Distributed ViewPoints: the anatomy of a consistency check", International Journal on Concurrent Engineering: Research & Applications, 2,3, CERA Institute, USA, pp. 209-222.

Easterbrook S., Nuseibeh B., 1996. "Using ViewPoints for Inconsistency Management", Software Engineering Journal, 11, 1, BCS/IEE Press, pp. 31-43.

Finkelstein A., Spanoudakis, G. & Till D., 1996. "Managing Interference", Joint Proceedings of the Sigsoft '96 Workshops – Specifications '96, ACM Press, pp. 172-174.

Finkelstein A., Gabbay D., Hunter A., Kramer J., & Nuseibeh B., 1994. "Inconsistency Handling In Multi-Perspective Specifications", IEEE Transactions on Software Engineering, 20, 8, pp. 569-578.

Guarino N., 1994. "The Ontological Level", Philosophy and the Cognitive Science, (eds) Casati R., Smith B. & White G., Vienna: Holder-Pichler-Tempsky.

Grueger T., 1993. "Toward Principles for the Design of Ontologies Used for Knowledge Sharing", International Workshop on Formal Ontology, Padova, Italy (also available as Technical Report KSL 93-04, Knowledge Systems Laboratory, Stanford University)

Heitmeyer C., Labaw B., Kiskis D., 1995. "Consistency Checking of SCR-Style Requirements Specifications", Proceedings of the 2nd International Symposium on Requirements Engineering (RE '95), IEEE CS Press, pp. 56-63.

LAS Inquiry Report, 1993. "Report of the Inquiry into the London Ambulance Service", Communications Directorate–South West Thames Regional Health Authority, ISBN-0-905133-70-6, London.

Leite J.C.S.P., Freeman P.A., 1991. "Requirements Validation through Specification Resolution", IEEE Transactions on Software Engineering, Vol. 12, No. 12, pp. 1253-1269.

Nissen H., Jeusfeld M., Jarke M., Zemanek G., Huber H., 1996. "Managing Multiple Requirements Perspectives with Metamodels", IEEE Software, March 1996, pp. 37-47.

Robinson W., Fickas S., 1994. "Supporting Multiple Perspective Requirements Engineering", Proceedings of the 1st International Conference on Requirements Engineering (ICRE 94), IEEE Computer Society Press, pp.206-215

Spanoudakis G., Constantopoulos P., 1995. "Integrating Specifications: A Similarity Reasoning Approach" Automated Software Engineering Journal, 2, 4, pp. 311-342.

Spanoudakis, G., Finkelstein, A., 1997. "Reconciling Requirements: a method for managing interference, inconsistency and conflict", Annals of Software Engineering (to appear in the Special Issue on Software Requirements Engineering)

Spanoudakis G., Finkelstein A., Till D., 1997. "Interference in Requirements Engineering: The Level of Ontological Overlap", Technical Report Series, TR-1997/01, ISSN 1364-4009, Department of Computer Science, City University.

van Lamsweerde A., 1996. "Divergent Views in Goal-Driven Requirements Engineering", Joint Proceedings of the Sigsoft ‘96 Workshops – Specifications ‘96, ACM Press, pp. 252-256.

Zave P., Jackson M., 1993. "Conjunction as Composition", ACM Transactions on Software Engineering and Methodology, Vol. 2, No. 4, pp. 379-411.