Ontology and Model Alignment as a Means for Requirements Validation

Leonid Kof
Fakultät für Informatik, Technische Universität München,
Boltzmannstr. 3, D-85748, Garching bei München, Germany,
kof@in.tum.de

Ricardo Gacitua, Mark Rouncefield, and Pete Sawyer
Computing Department, InfoLab21, South Drive,
Lancaster University, Lancaster, UK, LA1 4WA,
{r.gacitua | m.rouncefield | p.sawyer}@lancaster.ac.uk

***Abstract*—This paper reports on work that is investigating the application of ontology engineering and natural language processing to software engineering. Our focus is the transition from requirements to design which remains one of the main challenges in software engineering. A key reason for why this is so challenging is that the vast majority of requirements documents are informal, written in natural language, whereas the final goal (code) is formal.**

System models, as an intermediate step between the requirements and code, help understand requirements. Even a seemingly precise requirements document typically contains a lot of inconsistencies and omissions, which become visible when we model the system. Our hypothesis is that these inconsistencies become apparent when we compare the project-specific model with a generic model of the application domain. To test our hypothesis, we need to transform natural language representations of requirements information into a form that facilitates comparison with a domain model. Naturally, we also need a domain model against which to compare and this presupposes a means to construct such models.

In the paper, we extract a conceptual model (an ontology) and a behavioural model from different sources. An ontology is generated from a generic domain description, and a project-specific model is generated from requirements documents. For ontology generation, natural language processing techniques are used to aid the construction. By comparing the resulting models, we validate both of them. When inconsistencies are found, we generate feedback for the analyst. The generated feedback was validated on a case study and has proven useful to improve both requirements documents and models.

I. Requirements Documents Suffer from Missing Information

At the beginning of every software project, some kind of requirements document is usually written. There are many different modeling notations that support the precise description of requirements and which support reasoning to help achieve completeness and consistency in the specified requirements. However, use of these notations is only feasible if they are intelligible to the documents' authors and to the stakeholders who are required to approve the documents' contents. In most cases, they are not and therefore, as the survey by Mich et al. shows [1], the great majority of requirements documents are written in natural language. As a consequence, most requirements documents are imprecise, incomplete, and inconsistent, because precision, completeness and consistency are extremely difficult to achieve using natural language as the main presentation means.

This work was supported by a fellowship within the Postdoc-Programme of the German Academic Exchange Service (DAAD) and by EPSRC grant EP/F069227/1 MaTREx.

Document authors are mostly unaware of incompleteness or inconsistencies of requirements documents. In software development, the later an error is found, the more expensive its correction [2]. Thus, it is one of the goals of requirements analysis, to find and to correct the defects of requirements documents. A practical way to detect errors in requirements documents is to convert informal specifications to models. In this case, errors in documents would lead to inconsistencies or omissions in models, and, due to more formal nature of models, inconsistencies and omissions are easier to detect in models than in textual documents. However, the automatic generation of models from textual requirements is possible only if the document authors are constrained to employ a controlled subset of (e.g.) English, and this constraint is generally not acceptable in requirements engineering. Nevertheless, techniques used in semantic computing have the potential to aid model generation and validation, even for requirements documents written using uncontrolled natural language.

In the presented paper, we introduce the following means of requirements validation: first, we extract domain-specific abstractions and relationships between them from a document or documents that are representative of the application domain. These are assembled into an ontology that acts as a conceptual model of the problem domain. We then take descriptions of the desired behaviour of the system-to-be, in the form of scenarios or use case descriptions, and construct a set of Message Sequence Charts (MSCs). The resulting models (ontology and MSCs) can then be checked for consistency. When inconsistencies are found, we generate feedback questions for the requirements analyst. These questions allow us to highlight deficiencies of the requirements document, to improve the document, and to validate both the requirements and the extracted models. This consistency check, as a means of validation and feedback generation, is the main contribution of the presented paper.

Outline: The remainder of the paper is organised as follows: Section II presents an existing approach to ontology extraction. Sections III and IV are the technical core of the paper: Section III presents the proposed method of ontology and model alignment, and Section IV the evaluation of the method.

Then, Section V presents the lessons learned in the case study. Finally, Sections VI and VII present the related work and the summary of the paper.

For the remainder of the paper we use the following terminology: A *scenario* is a sequence of natural language sentences. An *MSC* consists of a set of *actors*, a sequence of *messages* sent and received by these actors, and a sequence of *assertions* (statements about actors or their states) interleaved with the message sequence.

II. Extraction of a Domain Ontology

An ontology represents the most fundamental knowledge pertinent to the application domain, namely the concepts constituting the domain and the relationships between them. An ontology should serve as a common language for all stakeholders involved in a project [3]. In the ideal case, an ontology should represent the knowledge that is not specific to a single project, but provides background for the whole application domain. This would allow the alignment of project-specific knowledge with general domain knowledge (see Section III).

In the presented work, we assume that an ontology consists of three elements: a set of concepts, a classification of these concepts (taxonomy), and a set of non-taxonomic relationships between the concepts. Accordingly, we perform the following steps to extract a domain-specific ontology: First, we extract the domain-specific concepts from a corpus or a large generic domain-specific document. Then, we classify the extracted concepts and produce a taxonomy. Finally, we look for non-taxonomic relationships between the concepts. Each of these steps is presented below in detail.

1) Concept extraction: In order to extract domain-specific concepts from the document, we use the extraction techniques implemented in OntoLancs [4]. Two techniques are used for concept extraction:

- **Frequency profiling:** There exist linguistic corpora, like the British National Corpus [5] that document the everyday usage of language. If the frequency of some term in a domain specific document significantly deviates form the frequency of the same term in everyday usage, this can be an indicator that this particular term is domain specific [6]. Additionally to pure frequency profiling, we can use part-of-speech (POS) tagging to filter the results. In this case we obtain the specified part of speech only (nouns, adjectives, ...) as suggested terms. OntoLancs provides a user interface to sort all the extracted concepts by their relevance (=frequency deviation) and to decide, which of the concepts should become a part of the domain ontology.
- **Relevance-driven abstraction identification:** Corpus-based frequency profiling technique works well for concepts that are signified by single words, but not for compound terms. In order to adapt frequency profiling to compound terms, a new technique called RAI (Relevance-driven abstraction identification) was recently added to OntoLancs [7]. The RAI algorithm comprises the following steps:
  1) Each word in the domain text is annotated with a part-of-speech tag.
  2) The set of words is filtered to remove stop words.
  3) The remaining words are lemmatized to reduce them to their dictionary form, i.e. to collapse inflected forms of words to a base form or lemma.
  4) Each word is assigned a log-likelihood value by applying the corpus-based frequency profiling.
  5) Syntactic patterns are applied to the text to identify multi-words term.
  6) A significance score is derived from the frequency profiling.
  7) The set of terms are ranked according to their significance score.

  As in the case of frequency profiling, the user decides which of the suggested terms are relevant for the domain ontology.

2) Concept classification: Based on the lexical form of the extracted concepts, OntoLancs classifies them and provides a taxonomy: A concept B is assumed to be a subordinate concept of A ("every instance of B is also an instance of A"), if the lexical form of A is a part (substring) of the lexical form of B. For example, in this manner, OntoLancs suggests that "potential overseas applicant" is a subordinate concept of "overseas applicant", which, in turn, is a subordinate concept of "applicant". The taxonomy proposed by OntoLancs can be reviewed and corrected by a human analyst.

3) Non-taxonomic relationships: In order to extract non-taxonomic relationships, we consider the occurrences of previously extracted concepts in the same sentence. The basic assumption is that, whenever two terms occur in the same sentence, there is a non-taxonomic relationship between them. For example, if "applicant" and "admission officer" are previously extracted concepts, we infer a relationship between them from the sentence "The applicant sends all the documents to the admission officer". For compound sentences, we assume a non-taxonomic relationship even if the concepts occur in different parts of the sentence. For example, under the assumption that "applicant", "admission office", and "confirmation" are previously extracted concepts, we extract three relationships from the sentence "Whenever the applicant sends the documents to the admission office, he/she receives a confirmation": relationship between "applicant" and "confirmation", "applicant" and "admission office", and between "admission office" and "confirmation". This, rather liberal, definition of a non-taxonomic relationship allows us to use the same technique to extract different types of relationships.

III. Alignment of Ontology and Models

The basic idea of our approach is fairly simple: first, we construct an application model ontology using OntoLancs. Then, we cut out the parts of the ontology that are relevant for the scope of the MSCs: we assume that the taxonomy contains at least three separate branches, with one branch containing actors, one containing messages, and one containing assertions.

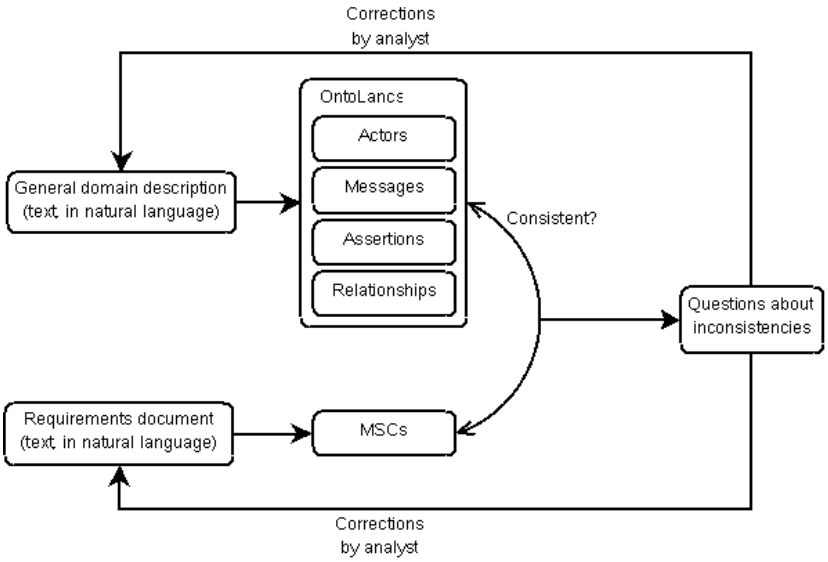

Fig. 1. Analysis procedure

Independently from the ontology extraction, we formalise application scenarios in Message Sequence Charts (MSCs). Formalisation of behaviour in MSCs provides more information than is explicitly stated in the text, so such formalisation allows us to go beyond pure comparison of ontologies extracted from generic domain documentation and requirements documents. When the behaviour is formalised, we check the consistency between the ontology and the MSCs. If any inconsistencies are found, we generate feedback for the requirements analyst. This procedure is illustrated in Figure 1. To extract different types of concepts (actors, messages, and assertions), we manually classify the concept classes suggested by OntoLancs.

The consistency check, which is the core of the presented work, works in two phases:

- **Sets of concepts:** The consistency check extracts the sets of actors, messages, and assertions from the MSCs. These sets are compared with the sets extracted by OntoLancs from the domain documents. In the ideal case, these sets should coincide. For every concept present in the sets extracted by OntoLancs and accepted by the analyst, but not present in the MSCs, the consistency check generates a feedback message like "Actor/Message/Assertion X is present in the ontology but not present in MSCs". Concepts that are present in the MSCs but not extracted by OntoLancs are treated similarly.
- **Relationships**: Every assertion in the MSC implies several relationships: it implies a relationship between the text of the assertion and every actor involved in the assertion. Similarly, every message implies three relationships: a communication path between the message sender and message receiver, and relationships between the message content and the message sender or receiver.
  The consistency check compares these relationships, implied by the MSCs, with the relationships extracted from the generic document. If a relationship is implied by some MSC, but not extracted from the generic document, the consistency check generates three types of feedback messages:
  - "Communication path from actor X to actor Y is not explicitly specified in the ontology"
  - "Relationship between message X and its sender/receiver is not explicitly specified in the ontology"
  - "Relationship between assertion X and actor Y is not explicitly specified in the ontology"

  If some relationship is present in the ontology but not in the MSCs, the consistency check generates a feedback message like "Relationship between concepts X and Y is present in the ontology, but not in the MSCs".
  In order that the analyst is not overwhelmed with too many feedback messages, the messages are generated if and only if both concepts involved in the relationship are present in the MSCs and in the ontology. Otherwise, the corresponding inconsistency is already addressed by the feedback messages about sets of concepts.

The generated feedback messages are presented to the requirements analysts. They can be used in many ways: to improve the ontology and the MSCs, or to rewrite the documents that lead to inconsistent ontologies/MSCs. In this way, these feedback messages serve to validate both the requirements and the models.

IV. Evaluation

The presented approach was evaluated on a case study, presented below in Section IV-A. In the case study, we compared an ontology extracted from a background document describing the application domain, and a set of MSCs, constructed

manually on the basis of five application scenarios. Here, it is important to emphasise that the application scenarios used to construct MSCs were not included in the document used for ontology extraction, so the ontology and the MSCs really specified different system aspects. Ontology extraction and MSC construction are presented in Section IV-B. Then, based on the consistency check of the ontology and the MSCs, we generated feedback messages and reviewed our findings in two interactive session with an application domain expert. This part of the evaluation is presented in Section IV-C.

A. Case Study: Postgraduate Admission Portal

A specification of a postgraduate admissions portal for a UK university was used as a case study in the presented work. For ontology extraction, we used a generic document about the application domain, approx. 100 pages long.

To construct MSCs, we used five scenarios that were provided independently from the generic domain document. The scenarios specify the desired system behaviour in exemplarily situations, like this:

- Miss X, having completed her undergraduate studies in the PRC, is interested in studying a professionally-accredited postgraduate finance course in the UK.
- Mrs Y, a management school (MS) faculty admissions officer (FAO), logs on to the system and sees an enquiry from Miss X about local transport options.
- The enquiry also requests that Miss X can book a chat session with somebody about student accommodation and funding options.
- ...

MSCs were chosen as representation means, because they allow the specification of both communication between different actors, as well as statements concerning single actors.

When we construct MSCs for such scenarios, we actually provide more information than contained in the text: For example, in the above scenario, it is our interpretation of the text that the third sentence ("The enquiry also requests...") is represented as a message from "system" to "Mrs Y", as "system" is not an explicitly specified actor. Such interpretations allow us to model more information than is actually contained in the text. This, in turn, allows us to go beyond pure comparison of ontologies (one extracted from the generic document, and one extracted from scenarios) and to see how the scenarios can be interpreted in software development.

B. Ontology Extraction and Model Construction

In order to extract a domain-specific ontology, we applied OntoLancs to the background document. This resulted in the extraction of different concept classes. For the purpose of the alignment of this ontology with MSCs, the following concept classes were considered relevant:

- MSC Actors:
  - applicant (prospective applicant, enquirer, UK applicant, EU applicant, overseas applicant, ...)
  - admission staff (departmental admission staff, admission tutor, admission officer, ...)
  - accommodation service
  - PGAO (=postgraduate admission office)
- MSC Messages (information or documents that are exchanged between actors):
  - application, postgraduate application
  - admission information, admission criteria
  - accommodation information, department information, course information
  - offer, rejection
  - enquiry (department enquiry, course enquiry, ...)
  - document (supporting document), documentation (supporting documentation)
  - application form

There were no concepts extracted that could be interpreted as assertions. For this reason, assertions were not considered in the subsequent consistency checks.

To construct the MSCs from scenarios, we manually interpreted every scenario. When constructing the MSCs, we did not have access to a domain expert, and this constraint makes our case study setting similar to real software projects, where the availability of domain experts is typically limited too. Nevertheless, from our understanding of the application domain, the constructed MSCs represent a valid interpretation of the scenarios.

C. Findings of the Case Study

In order to identify the inconsistencies between the ontology and MSCs, we aligned them, as described in Section III. The ontology extracted from the domain document contained no assertions, so assertions in the MSCs were not taken into account for the alignment. When aligning the ontology with the models, we had to take into account that the same concept can have different lexical forms. For example, we can have a sentence like "The applicant is rejected due to insufficient level of English" in the scenario, which results in a message "rejected due to insufficient level of English" from the "admission officer" to the "applicant" in an MSC. On the other side, we had just the "reject" concept in the ontology. To match the above two concepts, we assumed that a message in the MSC matches a message from the ontology, if the message name, as encoded in the ontology, is a substring of the message representation in the MSC. This allows us to state that the "reject" concept in the ontology matches the "reject due to insufficient level of English" message in the MSC. For actors, however, we required the exact name identity in the ontology and in the MSC.

With this procedure, 95 inconsistencies between the ontology and the MSCs were detected:

- 16 actors present in the ontology, but not in the MSCs
- 8 actors present in the MSCs but not in the ontology
- 12 messages present in the ontology but not in the MSCs
- 59 messages present in the MSCs but not in the ontology

We converted the feedback messages about detected inconsistencies into questions of the form "Should actor/message X, absent in the ontology/MSCs, be included in it?". An

application domain expert answered these questions in two sessions, two hours each. The first comment of the domain expert was that the abstraction levels of the document used for ontology extraction and for the scenarios/MSCs were different, so we could not expect perfect consistency between them.

When we went through the generated questions, it turned out that the questions like "Should message X be present in the ontology?" could not be answered directly in the most cases. We had to distinguish the lexical level ("Should the exact lexical form of message X be present in the ontology?") and the semantic level ("Should the concept conveyed by message X be present in the ontology?").

At the lexical level, it turned out that most of the detected inconsistencies are spurious:

- from 16 actors present in the ontology, but not in the MSCs, 13 should be present in the MSCs (**81%**)
- from 8 actors present in the MSCs but not in the ontology, 2 should be present in the ontology (**25%**)
- from 12 messages present in the ontology but not in the MSCs, 6 should be present in the MSCs (**50%**)
- from 59 messages present in the MSCs but not in the ontology, 6 should be present in the ontology (**10%**)

This implies the overall precision of 27/95≈**28%** for questions about messages and actors.

At the semantic level, however, most generated questions were relevant: they showed connections between the generic concepts contained in the domain document and their concrete realisation in MSCs. Here, we could not obtain a 1:1 mapping between ontology- and MSC-concepts, so we could not measure the precision of the question generation. We simply rely on the statement made by the domain expert, that, at the semantic level, most generated questions were relevant.

Apart from the questions concerning the sets of actors/messages, we generated questions about communication paths: in MSCs, every message implies a communication path between its sender and its receiver. We expected that, in the generic domain document, such communication paths are signified by sentences in which the message sender and message receiver co-occur. It turned out that this was not always the case: We identified six communications paths that were present in the MSCs, but not signified in the domain document. Our domain expert found that all these paths should be present in the domain document.

To summarise, even though the presented approach cannot enforce complete consistency between domain ontologies and MSCs, it turned out to help bridging the gap between a generic domain document and application scenarios. This means, in particular, that we facilitate understanding of the application domain by the requirements analyst.

V. Lessons Learned

Conducting the case study has revealed a number of important lessons that we shall incorporate into our future work, and which, we believe, have wider relevance for applications of semantic computing to requirements engineering. The primary lesson is that different requirements documents have different purposes. These purposes can affect how inferred semantics needs to be interpreted. The background document chosen from which to extract the domain ontology was produced not only to provide a technology-agnostic description of the domain of postgraduate student admissions, but also to win support from a review board comprising senior academics and university managers to invest in development of new business processes and support systems. The implications of this are: (1) different levels of abstraction, (2) reliance on implicit knowledge, and (3) influence of politics on the documents. These points are discussed below in detail.

Levels of abstraction. There is an inevitable mismatch between domain concepts represented in the ontology and solution concepts represented in the scenarios used to derive user requirements. The most obvious of these derives from the ubiquity of the "system" actor in the scenarios, which contrasts with its absence from the ontology. Explicitly or implicitly, most interaction in the scenarios takes place between user actors and the system. Thus, many of the messages that appear in the MSCs simply do not appear in the ontology.

Implicit knowledge. Both the document used to extract the domain ontology and the scenarios rely on implicit knowledge for their interpretation. This manifests itself at several levels. An example is illustrated in the scenario in Section IV-A which contains an implicit precondition that Miss X has logged an enquiry about the availability of professionally-accredited finance courses. More subtle examples also occur, and these are even harder to discover. In interpreting the role of the actor Miss X, we normalized the signifier "Miss X" to the underlying concept of overseas applicant. Validation of this step with the domain expert revealed that overseas applicant was highly nuanced. Depending on context overseas applicant indicated either:

- Fee status - how much the applicant would pay in tuition fees, and therefore an attribute of the applicant actor;
- A market segment - different marketing strategies are used to recruit UK students, students from elsewhere in the EU, and for students from other parts of the world;
- A strategic goal - the university aims to improve the rate of conversion of overseas applicants to students who subsequently enrol.

While these alternative interpretations didn't impact on the structure of Miss X's interactions in the scenario, the semantics of how those interactions handled may be contingent on the interpretation. Matching the normalized actor overseas applicant from the scenarios with occurrences of *overseas applicant* in the domain document may be misleading, since in reality they may represent different interpretations.

Politics. The document used to extract the domain ontology was not neutral; it reflected the agenda of its authors, which was to promote investment in process change and, more subtly, to promote one of several possible process configurations. One result of this could have been, for example, to suppress other alternatives or the roles or even existence of process actors.

Sometimes these factors can be mitigated by, for example, careful selection of the source document(s) from which to

generate the domain ontology. Others are more problematic, however. In particular, implicit or suppressed description is very hard to detect automatically and may result in poor results. The factors highlight the importance of recognizing that the difficulties facing semantic computing when applied to the requirements engineering process stem not only from the properties of language, but also from context and the purpose for which they are written.

VI. Related Work

Approaches related to our work can be roughly subdivided into areas of comparison of different versions of the same basic model, and comparison of different views representing the same system. The approaches to compare different versions of the same model were introduced, for example, by Antoun et al. [8], Bendix and Emanuelsson [9], and Bartelt [10]. These approaches have a different focus from our work, as we compare different views on the same system.

Theoretical basis for comparison of different views was provided by Brunet et al. [11], Kelter and Schmidt [12], and Rubin et al. [13]. These approaches either state requirements for approaches to model comparison, or provide generic frameworks to define rules for model comparison. However, they do not generate feedback for requirements analysts.

Approaches that compare different views on the same system and generate feedback are most close to our work. The approach by Sabetzadeh et al. [14] compares models produced by different stakeholders and provides an integrated model. If inconsistencies are detected between the source model, they are explicitly documented in the resulting integrated model. Van Lamsweerde et al. [15] and Uchitel et al. [16] compare different views on the behaviour of the system. They take a set of MSCs, generate new MSCs that potentially represent further possible system behaviour, and ask the user if the newly generated MSCs really represent allowed system behaviour. The input for the above approaches is a set of formal models. In this sense, these approaches are complementary to our work, as we ground the models in requirements documents, written in natural language, and validate initial formal models.

VII. Conclusion

In our work, we are trying to mitigate the problems caused by the ubiquity of natural language as the medium for expressing user requirements, using techniques from semantic computing, in particular ontology engineering and natural language processing (NLP). NLP is not powerful enough for a full-fledged semantic analysis, so we implement techniques to assist human analysts with formalisation and validation of requirements through the construction of models from the requirements texts. By comparing these models with a generic domain ontology, we are able to help the analyst uncover information missing from the requirements documents.

Both requirements modelling and ontology extraction are non-trivial tasks and the presented approach does not claim to solve all their problems. However, it provides important links between behaviour models and ontologies:

- Given an ontology and behaviour models, it checks for consistency between them, and, on this basis, generates feedback for analysts.
- Consistency check, in turn, allows to validate and to improve both textual documents and models.

To summarise, the proposed approach provides valuable feedback on model quality and consistency and can be successfully applied in software engineering.

Acknowledgments

We are grateful to Paul Holland for his help in the evaluation and for his valuable feedback on our method.

References

[1] L. Mich, M. Franch, and P. Novi Inverardi, "Market research on requirements analysis using linguistic tools," *Requirements Engineering*, vol. 9, no. 1, pp. 40–56, 2004.

[2] B. W. Boehm, *Software Engineering Economics*. Prentice-Hall, 1981.

[3] H.-J. Happel and S. Seedorf, "Applications of ontologies in software engineering," in *International Workshop on Semantic Web Enabled Software Engineering (SWESE'06)*, Athens, USA, November 2006. [Online]. Available: http://fparreiras/papers/AppOntoSE.pdf

[4] R. Gacitua, P. Sawyer, and P. Rayson, "A flexible framework to experiment with ontology learning techniques," *Knowledge-Based Systems*, vol. 21, no. 3, pp. 192 – 199, 2008, aI 2007, The 27th SGAI International Conference on Artificial Intelligence. [Online]. Available: http://www.sciencedirect.com/science/article/B6V0P-4R6B2HC-F/2/549a3ec9b4320c97e5a572e30be5fa97

[5] G. Aston and L. Burnard, *The BNC Handbook: Exploring the British National Corpus with SARA*. Edinburgh University Press, 1998.

[6] P. Sawyer, P. Rayson, and K. Cosh, "Shallow knowledge as an aid to deep understanding in early phase requirements engineering," *IEEE Trans. Softw. Eng.*, vol. 31, no. 11, pp. 969–981, 2005.

[7] R. Gacitua, P. Sawyer, and V. Gervasi, "On the effectiveness of abstractions identification in requirement engineering," in *Proceedings of the 18th IEEE International Conference on Requirements Engineering*. Washington, DC, USA: IEEE Computer Society, 2010, to appear.

[8] M. Abi-Antoun, J. Aldrich, N. Nahas, B. Schmerl, and D. Garlan, "Differencing and merging of architectural views," in *ASE '06: Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 47–58.

[9] L. Bendix and P. Emanuelsson, "Requirements for practical model merge — an industrial perspective," in *MODELS '09: Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 167–180.

[10] C. Bartelt, "Consistence preserving model merge in collaborative development processes," in *CVSM '08: Proceedings of the 2008 international workshop on Comparison and versioning of software models*. New York, NY, USA: ACM, 2008, pp. 13–18.

[11] G. Brunet, M. Chechik, S. Easterbrook, S. Nejati, N. Niu, and M. Sabetzadeh, "A manifesto for model merging," in *GaMMa '06: Proceedings of the 2006 international workshop on Global integrated model management*. New York, NY, USA: ACM, 2006, pp. 5–12.

[12] U. Kelter and M. Schmidt, "Comparing state machines," in *CVSM '2008: International Workshop on Comparison and versioning of software models*. New York, NY, USA: ACM, 2008, pp. 1–6.

[13] J. Rubin, M. Chechik, and S. M. Easterbrook, "Declarative approach for model composition," in *MiSE '08: Proceedings of the 2008 international workshop on Models in software engineering*. New York, NY, USA: ACM, 2008, pp. 7–14.

[14] M. Sabetzadeh and S. Easterbrook, "View merging in the presence of incompleteness and inconsistency," *Requir. Eng.*, vol. 11, no. 3, pp. 174–193, 2006.

[15] C. Damas, B. Lambeau, and A. van Lamsweerde, "Scenarios, goals, and state machines: a win-win partnership for model synthesis," in *SIGSOFT FSE*, 2006, pp. 197–207.

[16] S. Uchitel, J. Kramer, and J. Magee, "Synthesis of Behavioral Models from Scenarios," *IEEE Trans. Softw. Eng.*, vol. 29, no. 2, pp. 99–115, 2003.