

Faculty of Informatics

Faculty of Informatics - Papers

University of Wollongong

Year 2006

Viewpoints Merging via Incrementally Elicited Ranked Structures

A. Ghose*

Q. Lin†

*University of Wollongong, aditya@uow.edu.au

†University of Wollongong, qlin@uow.edu.au

This article was originally published as: Ghose, A & Lin, Q, Viewpoints Merging via Incrementally Elicited Ranked Structures, Sixth International Conference on Quality Software (QSIC 2006), Beijing, China, October 2006, 141-150. Copyright IEEE 2006.

This paper is posted at Research Online.

<http://ro.uow.edu.au/infopapers/490>

Viewpoints Merging via Incrementally Elicited Ranked Structures

Aditya Ghose and Qiuming Lin
Decision Systems Lab
School of IT and Computer Science
University of Wollongong
NSW 2452
Australia
aditya@uow.edu.au

Abstract

Handling inconsistent requirements specifications remains a difficult challenge for the requirements engineering community. This paper seeks to apply techniques developed in the belief merging literature within AI to this problem. The application is non-trivial, since many belief merging operations cannot be directly applied, but must be modified to make them usable in practical settings. We address the problem of merging state model viewpoints, and improve on previous work by Chechik and Easterbrook. We develop a variant of the belief merging framework developed by Meyer et al, which we refer to as the framework of incrementally elicited ranked structures. We show that viewpoint merging within this framework is relatively easy and provides meaningful results.

1. Introduction

The problem of multi-perspective specifications and dealing with potentially inconsistent viewpoints has posed a major challenge to the requirements engineering community. Approaches to dealing with inconsistency in requirements have a relatively long history. Balzer [1] introduced the notion of *pollution markers* as an approach to tolerating and managing inconsistency in specifications. Tsai proposed the use of non-monotonic logics in resolving inconsistencies in specifications [13] while similar ideas were also explored by Ryan [12]. The *Viewpoints* framework [5] [10] [4] [3] supports multi-perspective development (with multiple sets of stakeholders) by allowing explicit “viewpoints” which hold partial specifications, described and developed using different representation schemes and development strategies. Individual viewpoints are required to be internally consistent while inconsistencies arising between

pairs of distinct viewpoints (the authors suggest translation into a uniform logical language for detecting inconsistencies) are removed by invoking meta-level inconsistency handling rules. Lamsweerde *et al* [14] have explored a wide range of categories of inconsistency in the context of the KAOS framework. Wiels and Easterbrook [15] have defined evolution and inconsistency handling techniques based on category theory, while Nuseibeh and Russo [11] have used abductive logic programming. Heitmeyer et al [6] have defined inconsistency handling techniques in the context of tabular notations. Hunter and Nuseibeh [7] have defined a framework for representing specifications using a logic with a paraconsistent flavour.

Research on *belief merging* within Artificial Intelligence has addressed the problem of merging the (potentially inconsistent) beliefs of a society of agents, who might need to arrive at an agreement in order to cooperate to achieve their goals, or to support social choice processes (such as elections, committee decisions etc.). A substantial body of literature exists in the area, drawing on diverse disciplines such as social choice theory in economics. We do not survey this literature here due to space constraints. The objective of this research is to explore ways in which results from the belief merging area might be deployed to address the viewpoints merging problem. Our results suggest that the connections are very rich and that novel viewpoints merging approaches can be developed by drawing on belief merging techniques. However, the application of belief merging techniques is non-trivial, and substantial changes must be made to make them viable for application in viewpoint merging problems.

This paper addresses the problem of merging state model viewpoints. We take as our starting point the work of Easterbrook and Chechik (a large number of publications on their framework exist, but [2] is a representative example). They use an underlying multi-valued logic and multi-valued model checker to determine sources of inconsistency in dis-

tinct viewpoints. For instance, in a setting with two stakeholders and a 4-valued logic consisting of the truth values TT, TF, FT and FF, elements of a model that evaluate to TT or FF suggest agreement amongst the stakeholders (that the corresponding elements are true or false, respectively) while those that evaluate to TF or FT indicate sources of disagreement. Their framework is largely useful in highlighting sources of disagreement or inconsistency, but provide no guidance on how these might be resolved.

In this paper, we modify Meyer's work on belief merging [9] to obtain a framework for viewpoints merging with *incrementally elicited ranked structures*. We have implemented a tool to support the process, and present a simple case study in which this tool is applied. The tool incorporates support for model checking on alternative merged viewpoints to ensure that only models which satisfy the relevant properties are adopted as outcomes of the process. Unlike Easterbrook and Chechik, our approach suggests actual merged outcomes. An added benefit is the ability to formally establish correctness of the merging process via reference to a set of commonly agreed upon *rationality postulates* for merging [9]. We do not describe this aspect of our work any further due to space constraints.

2. Background Belief Merging

We base our work on Meyer's approach to belief merging [9], where each agent's belief state is represented by a preference ordering on models (or valuations, i.e., a mapping of each propositional letter to a boolean truth value). Meyer's uses a specific instance of these orderings, with each model/valuation being mapped to a natural number. We shall assume a propositional language L , U is the set of interpretations of L and $M(\alpha)$ is the set of models of $\alpha \in L$. We shall use Φ to denote an epistemic state and ϕ to denote the knowledge base associated with Φ . We let x_n denote the list containing n version of x . The length of a list l is denoted by $|l|$.

An epistemic state Φ is a function from U to the set of natural numbers. Given an epistemic state Φ , the knowledge base associated with Φ , denoted by ϕ_Φ , is some $\phi \in L$ such that $M(\phi) = \{u \mid \Phi(u) = 0\}$.

Epistemic states allow us to represent preference orderings on valuation (or models). Valuations which receive a rank of 0 are the most preferred, while those that get a rank of 1 are the next most preferred, and so on. Some numbers may have no valuations assigned to them (i.e. there may be empty ranks) suggesting that the relative distance between ranks can play a role in the specification of preference.

An *epistemic list* $E = [\Phi_1^E, \dots, \Phi_{|E|}^E]$ is a non-empty finite list of epistemic states. Each element of an epistemic list is an epistemic state representing the beliefs of an agent in the collection of agents whose beliefs must be merged.

For any epistemic state Φ , let

$$\min(\Phi) = \min\{\Phi(u) \mid u \in U\},$$

let

$$\max(\Phi) = \max\{\Phi(u) \mid u \in U\},$$

and for an epistemic list E , let

$$\max(E) = \max\{\max(\Phi_i^E) \mid 1 \leq i \leq |E|\}.$$

For an epistemic list E and $u \in U$, let $\min^E(u) = \min\{\Phi_i^E(u) \mid 1 \leq i \leq |E|\}$ and let $\max^E(u) = \max\{\Phi_i^E(u) \mid 1 \leq i \leq |E|\}$. $seq(E)$ denotes the set of all sequences of length $|E|$ of natural numbers, ranging from 0 to $\max(E)$. We denote by $seq \leq (E)$ the subset of $seq(E)$ of all sequences that are in non-decreasing order, and $seq \geq (E)$ the subset of $seq(E)$ of all sequences that are in non-increasing order. For $u \in U$, we let $s^E(u)$ be the sequences containing the natural numbers $\Phi_1^E(u), \dots, \Phi_{|E|}^E(u)$ in that order, we let $s_{\leq}^E(u)$ be the sequence $s^E(u)$ in non-decreasing order, and we let $s_{\geq}^E(u)$ be the sequence $s^E(u)$ in non-increasing order. Obviously $s^E(u) \in seq(E)$, $s_{\leq}^E(u) \in seq_{\leq}(E)$ and $s_{\geq}^E(u) \in seq_{\geq}(E)$. $s_i^E(u)$, $s_i^{(E, \leq)}$ and $s_i^{(E, \geq)}$ denote the i -th digit in $s^E(u)$, $s_{\leq}^E(u)$ and $s_{\geq}^E(u)$ respectively. Given any set seq of finite sequences of natural numbers and a total preorder \sqsubseteq on seq , we define the function $\Omega_{\sqsubseteq}^{seq} : seq \rightarrow \{0, \dots, |seq| - 1\}$ by assigning consecutive natural numbers to the elements of seq in the order imposed by \sqsubseteq , starting by assigning 0 to the elements lowest down in \sqsubseteq .

In the following, we will review some of Meyer's merging operators. In particular, we will review three specific operators: Δ_{min} , Δ_{max} and Δ_{Σ} . Meyer defines several others, but these three form a representative subset. Δ_{min} and Δ_{max} are examples of arbitration operators while Δ_{Σ} is an example of a majority operator.

There are two steps in the construction of each merging operation. The first step is to assign the rank (natural number) to each model (or valuation). After completing this step, if *none* of the models have been assigned a value 0, then the second step is to perform an appropriate uniform subtraction of values, which is referred to as *normalization*. In cases where there are no models of rank 0 (suggesting that the agent's beliefs are inconsistent) we normalize by shifting all of the ranks down, while maintaining their relative order and distance, but ensuring that the set of models at rank 0 are non-empty.

We consider the idea of an arbitration operation in which we take as many different viewpoints as possible from all the stakeholders into account. We will discuss two arbitration operators, the first of which is the Δ_{min} merging operator.

Definition 2.0.1. If E contains a single epistemic state Φ , let $\Phi_{min}^E = \Phi$. If not, let $\Phi_{min}^E(u) = 2\min^E(u)$ if $\Phi_i^E(u) = \Phi_j^E(u) \forall i, j \in \{1, \dots, |E|\}$ and $\Phi_{min}^E(u) = 2\min^E(u) + 1$ otherwise. Then $\Delta_{min}(E)(u) = \Phi_{min}^E(u) - \min(\Phi_{min}^E)$.

The Δ_{min} operator involves the following steps. Identify the models which are agreed to by all epistemic states as being the most preferred, and take them to be the most preferred model in the resulting epistemic state from the merging operation (assigning them the rank of 0). The models on the next level of preference are those deemed to be the most preferred by at least one epistemic state. The models on the next level of preference are considered to be the ones that are deemed to be the second most preferred by all the epistemic states and the models regarded as the second most preferred by at least one epistemic state are on the following level of preference. The above process is repeated until all levels of preference for all the epistemic states have been treated. The idea of Δ_{min} is to find the minimum preferred rank given to a model by any of the epistemic states and then to normalize the rank. The normalized rank is assigned as the new preference rank to the model.

Definition 2.0.2. Let $\Phi_{max}^E(u) = \max^E(u)$. Then $\Delta_{max}(E)(u) = \Phi_{max}^E(u) - \min(\Phi_{max}^E)$.

The maximum preference rank assigned to a model by any of the epistemic states is taken as the preference rank to that model.

Majority operators take the viewpoints of the majority stakeholders into account, i.e. it tries to minimize global dissatisfaction. The Δ_{Σ} merging operation is an example of a majority operation. For $s \in seq(E)$, let

$$sum^E(s) = \sum_{i=1}^{|E|} s_i$$

where s_i is the i th element of s .

Definition 2.0.3. Let $\Phi_{\Sigma}^E(u) = sum^E(s^E(u))$. Then $\Delta_{\Sigma}(E)(u) = \Phi_{\Sigma}^E(u) - \min(\Phi_{\Sigma}^E)$.

As before, the final sentence in the definition represents the normalization step. The idea of this operation is to obtain the new preference rank of the model by summing the preference ranks given by the different epistemic states (representing viewpoints of stakeholders) and then to normalize the ranks.

3. Merging via Incrementally Elicited Ranked Structures

3.1 Ranked structures

We introduce the idea of a *ranked structure* - a notion related to, but distinct from the notion of epistemic state used in Meyer's framework for belief merging. An epistemic state is intended to be a complete specification of an agent's epistemic state. Thus, it requires us to assign a rank to every possible state of affairs. In most non-trivial domains, the number of possible states of affairs is typically very large, and many of them, particularly at higher ranks (i.e. those that are less preferred), are often irrelevant to the discourse. In a realistic application domain, such as ours, we cannot conceivably have access to such a mapping. At best, we may ask agents (stakeholders) to rank the models elicited thus far. A ranked structure can thus be loosely viewed as being analogous to a partially specified epistemic state. There is another critical difference. In Meyer's approach to belief merging, we assume a commonly agreed upon language, relative to which models (or states of affairs) are conceived. In our context, each viewpoint comes with its own local vocabulary, relative to which a stakeholder specifies models. A global (common) vocabulary is eventually constructed via signature maps (described below), but this results in individual stakeholder models becoming incomplete (in general) relative to this global vocabulary. This represents another point of departure from the notion of an epistemic state. Meyer, Ghose and Chopra [8] have defined a syntactic approach to merging using ranked knowledge bases, but these are expressively equivalent to epistemic states, and hence inapplicable in our context for precisely the same reasons as those listed above.

Since viewpoints might be expressed in distinct vocabularies, we require *signature maps* to reconcile these, and assume that this is done by an analyst prior to the merging process. As in [2], we require that the mappings preserve type information, that every state and variable in the source models must map to a state and a variable in the merged model and that distinct names from the same source model are not mapped to the same name in the merged model.

3.2. Merging Algorithms

In this section, we present an algorithm for constructing our framework. It is a *lazy evaluation* approach, in that we are not obliged to obtain complete epistemic states a priori. The stakeholders start by giving their most preferred models, i.e models of preference rank 0. If the models presented by all the stakeholders are consistent, then we should retain all of these models and combine them into a single model. Otherwise, the stakeholders are required to supply their next most preferred models and the models are added to the ranked structure. They keep providing additional models until they reach an agreement, i.e. their models are

identical or consistent. Once an agreement is reached, sets of consistent models are combined into single models using the selected merging operator to determine the new preference ranks for them, and a new and merged ranked structure is thus formed.

The merged ranked structure is comprised of only combined models and the preference ranks assigned to them. The most preferred models of the merged ranked structure is first taken to check against the system properties set by the stakeholders using SMV model checker. If SMV returns a true value, this model will be the result model. If SMV returns a false value, then models of the next preference level of the merged ranked structure are model checked until a model is found to satisfy the properties and such model is the final outcome model. If no such model is found in the merged ranked structure, then we have to keep asking the stakeholders to give their models of the next preference level from where they previously reached an agreement, and repeat the above process to find a successful outcomes. The following algorithm of procedure **IncrementalMerge()** reflects this process.

Not all operators in Meyer's repertoire of merging operators lend themselves to an incremental elicitation approach to merging. We define below the *incrementality property* to circumscribe the set of merging operators that do lend themselves to incremental elicitation.

Definition 3.2.1. A merging operator is said to satisfy the incrementality property iff it is able to generate a complete merged epistemic state, up to rank($r-1$) if all epistemic states in input epistemic list are completely determined up to rank r , for $r \geq 1$.

It is easy to see that amongst Meyer's operators, only Δ_{min} , Δ_{max} , Δ_{Σ} and $\Delta_{R\Sigma}$ satisfy this property. In the rest of our discussion, we will only be interested in merging operators which satisfy the incrementality property. r represents the rank of the ranked structure and i represents to the number of stakeholders in all the algorithms described below.

procedure IncrementalMerge

inputs:

1. A set of partial ranked structures, $\{RS_i \mid i \in \text{STAKEHOLDERS}\}$
2. A merging operator OP and an associated function

artialMerge_{OP}

3. A set of CTL properties $PROP$

outputs:

1. A single partial ranked structure PM , represented as a sequence of sets $\langle S_0, S_1, \dots, S_r \rangle$, organized in ascending order of rank, where each set S_i contain models at rank i .
2. A model m

```

done := false
r := 0
repeat
  for each stakeholder  $i \in \text{STAKEHOLDERS}$ 
    elicit all models at rank  $r$  and place them in the set  $SM_r^i$ 
   $PM := \text{artialMerge}_{OP}(\{SM_k^j \mid j \in \text{STAKEHOLDER}, k \in \{0, \dots, r\}\})$ 
  if the  $(r - 1)$ th element of  $PM$  exists and is non-empty
    if there is a model  $m$  in  $PM_{r-1}$  that satisfies all properties in  $PROP$ 
      done := true
      return  $m, PM$ 
    else
      if there exists a model  $m' \in PM_r$  that satisfies all properties in  $PROP$ 
        done := true
        return  $m', PM$ 
      else
         $r := r+1$ 
  else
     $r := r+1$ 
until done

```

We define **PartialMerge_{OP}**() for instances where the merging operations under consideration are Δ_{min} , Δ_{max} and Δ_{Σ} in the following discussion. Thus **PartialMerge _{Δ_{min}}** $(x) \stackrel{def}{=} \text{PartialMerge}(x, \text{min})$, **PartialMerge _{Δ_{max}}** $(x) \stackrel{def}{=} \text{PartialMerge}(x, \text{max})$ and **PartialMerge _{Δ_{Σ}}** $(x) \stackrel{def}{=} \text{PartialMerge}(x, \Sigma)$ for all x . Other merging operations from [9] could also be supported by other instances of **PartialMerge_{OP}**(), but we do not elaborate them here, in the interests of brevity.

procedure PartialMerge()

inputs:

1. A set of partial ranked structures, one for each stakeholder. For a stakeholder i , a partial ranked structure is represented as sequence of sets $\langle SM_0^i, SM_1^i, \dots, SM_j^i \rangle$ where each set SM_j^i contains the models specified by stakeholder i at rank j
2. A function f , determined by the merging operator OP

outputs:

1. A single merged partial ranked structure $S : \langle S_0, S_1, \dots, S_k \rangle$

for each $m \in SM_j^i$ (for any i and any j)

$\text{CONS}(m) = \{\langle n, j \rangle \mid n \in SM_j^i, \text{ for any } i \text{ and any } j, \text{ s.t.}$

Consistent* $(m, n)\}$

$S_k := S_k \cup \{\langle n, k \rangle\}$ where $\langle n, k \rangle = \text{Combine}(\text{CONS}(m),$

$f)$

return S

Consistent* (m, n) is a test for the consistency of models m and n . Two models are consistent if the following rules are satisfied:

- If a variable is true in the state in one model, and the state is described in the second model, then the variable should be true or undefined in the state of the second model.
- If a variable is false in the state in one model, and the state is described in the second model, then the variable should be false or undefined in the state of the second model.
- If a transition between two states is described in one model, both of the states are described in the second model, then the transition should be described in the second model.

The following algorithm for function **Combine()** used in procedure **PartialMerge()** is for merging procedure involving the Δ_{max} and Δ_{Σ} operations.

function Combine(S, f)

inputs:

1. A set S of pairs of form $\langle m, l \rangle$, where m is a model and l is a rank such that $l \in \{0, \dots, r\}$. (All models referred to in S are guaranteed to be consistent).
2. A function f where $f = \max$ or $f = \Sigma$

outputs:

1. A combined model with its associated rank $\langle n, k \rangle$

$n := \{\}$
 $k := 0$
for $l = 0, \dots, r$ **do**
 $n := \mathbf{CombineModels}^*(n, m)$ where $\langle m, l \rangle \in S$
 $k := \begin{cases} \max(k, l) & \text{if } f = \max \\ k + l & \text{if } f = \Sigma \end{cases}$
return $\langle n, k \rangle$

The following algorithm for function **Combine()** using the Δ_{min} is slightly different from the above.

function Combine(S, f)

inputs:

1. A set S of pairs of form $\langle m, l \rangle$, where m is a model and l is a rank such that $l \in \{0, \dots, r\}$. (All models referred to in S are guaranteed to be consistent).
2. A function f where $f = \min$

outputs:

1. A combined model with its associated rank $\langle n, k \rangle$

$n := \{\}$
 $rank\text{-}set := \{\}$
for $l = 0, \dots, r$ **do**
 $n := \mathbf{CombineModels}^*(n, m)$ where $\langle m, l \rangle \in S$
 $rank\text{-}set := rank\text{-}set \cup \{l\}$
 $k := \begin{cases} 2l & \text{if } rank\text{-}set \text{ is a singleton and } rank\text{-}set = \{l\} \\ 2\min(rank\text{-}set) + 1 & \text{otherwise} \end{cases}$
return $\langle n, k \rangle$

The function **CombineModels***(m_1, m_2) takes two consistent models m_1 and m_2 as input and combines them into a single model m based on the following principles. We use $Var_m(s_i)$ to denote the set of variables that are assigned a value in state s_i in model m . We note that in a completely specified model, $Var_m(s_i)$ should be identical for each state s_i , but we allow for the possibility that users may incompletely specify a model.

We use $\langle s_i, s_j \rangle$ to denote a transition from s_i to s_j

- If a state s_i is defined in both models m_1 and m_2 , then s_i must be defined in m . $Var_m(s_i) = Var_{m_1}(s_i) \cup Var_{m_2}(s_i)$
- If a state s_i is defined in model m_1 but not in the model m_2 (or the reverse, without loss of generality), then state s_i must be defined in model m . $Var_m(s_i) = Var_{m_1}(s_i)$.
- If a transition $\langle s_i, s_j \rangle$ is defined in either m_1 or m_2 , $\langle s_i, s_j \rangle$ remains a transition in m .

We have described thus far a procedure for merging the incrementally elicited viewpoints of a fixed set of stakeholders. In real-life applications, the set of stakeholders may change - new stakeholders may join and existing ones may have. The approach described here has been extended to deal with these, but we do not describe this extension here due to space constraints.

4. A Simple Case Study

We have implemented the procedures described above and conducted three substantial case studies with the resulting tool, the State View Merge System (SVMS). For brevity, we present a pared down version of the simplest case study, involving requirements for a telephone system, below. The case study involved two individuals (we shall refer to them as Tom and Jerry in the following) who were given the following initial problem description in English.

The telephone handset has the functionality to receive a call. When it is idle, the receiver is replaced. When there is an incoming call, it is connected. If the incoming call is not answered, it will be disconnected and become idle again. It is also can be used to make a call.

The eventually agreed upon model was obliged to satisfy the following two properties:

1. If you are connected, you can replace the receiver. In CTL: $AG(\text{connected} \rightarrow EX(\sim\text{offhook}))$.
2. If you are dialing, you can receive an incoming call. In CTL: $AG((\text{offhook} \wedge \sim\text{connected}) \rightarrow EX(\text{connected}))$.

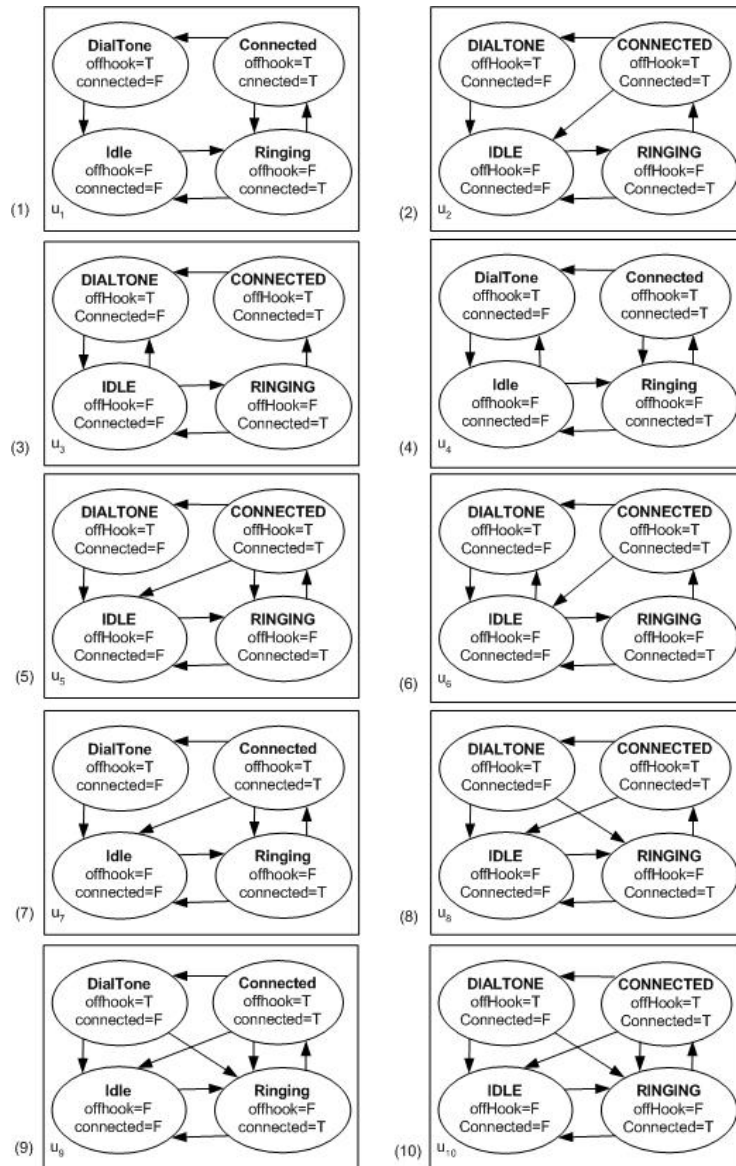


figure 1. All elicited viewpoints

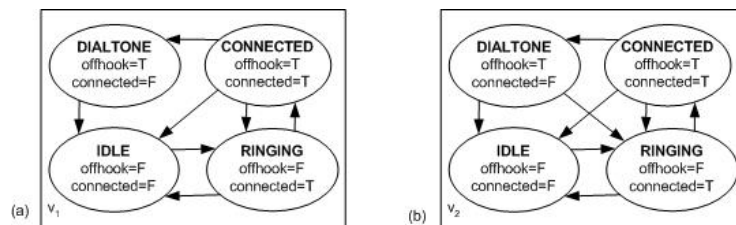


figure 2. Viewpoint obtained via merging

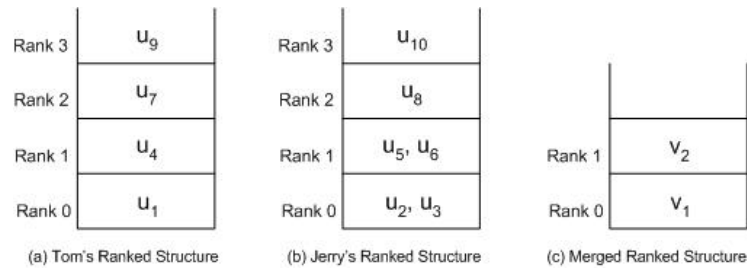


Figure 3. Ranked Structures

Tom and Jerry agreed to use the Δ_{max} merging operator and used SVMS to merge their initially inconsistent viewpoints (based on their diverging interpretation of the textual description given to them). The partially elicited ranked structures obtained for Tom and Jerry, as well as the eventual merged outcome, are represented in Figure 3 (which uses only model ID's - the corresponding models are presented in Figures 1 and 2). Note that Tom and Jerry use distinct vocabularies, which can be reconciled using signature maps as described above. The system found an initial agreement at rank 2 - Tom's model u_7 could be consistently combined with Jerry's model u_8 . However the SMV model checker (which the SVMS system interfaces with) found that the resulting combined model, v_1 , violates Property 2. The model elicitation process therefore continued, and an agreement was next identified between Tom and Jerry's models at rank 8 (u_9 and u_{10}) respectively. The combined model, v_2 , was found to satisfy both properties and was thus adopted as the final merged viewpoint.

5. Conclusion

In this paper, we have presented a novel approach to viewpoints merging that builds on prior work on belief merging in Artificial Intelligence. We started with Meyer's framework for belief merging [9] and modified it to obtain a more practically-grounded approach based on incrementally elicited ranked structures. We have described a case study using a tool that we have implemented for this approach. A key problem to address in future work is providing support for suggesting specific relaxations of models to stakeholders that might lead to earlier agreements.

References

- [1] R. Balzer. Tolerating inconsistency. In *Proceedings of the 13th Int'l Conference on Software Engineering*, pages 158–165, 1991.
- [2] S. Easterbrook and M. Chechik. A framework for multi-valued reasoning over inconsistency viewpoints. In *Proceedings of ICSE-2001*, pages 411–420.
- [3] S. Easterbrook, A. Finkelstein, J. Kramer, and B. Nuseibeh. Coordinating distributed viewpoints: The anatomy of a consistency check. In *Concurrent Engineering Research and Applications*, CERA Institute, West Bloomfield, USA, 1994.
- [4] A. Finkelstein, D. Gabbay, A. Hunter, J. Kramer, and B. Nuseibeh. Inconsistency handling in multi-perspective specifications. *IEEE Transactions on Software Engineering*, 20(8), 1994.
- [5] A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke. Viewpoints: A framework for integrating multiple perspectives in system development. *International Journal of Software Engineering and Knowledge Engineering*, 2(1):31–58, 1992.
- [6] C. L. Heitmeyer, R. D. Jeffords, and B. G. Labaw. Automated consistency checking of requirements specifications. *ACM Transactions on Software Engineering and Methodology*, 5(3):231–261, 1996.
- [7] A. Hunter and B. Nuseibeh. Analyzing inconsistent specifications. In *Proceedings of the Third IEEE International Symposium on Requirements Engineering*, pages 78–86, 1997.
- [8] T. Meyer, A. K. Ghose, and S. Chopra. Syntactic representations of semantic merging operations. In *Proceedings of the*

IJCAI 2001 Workshop on Inconsistency in Data and Knowledge, Seattle, USA, 2001.

- [9] T. A. Meyer. On the semantics of combining operations. *Journal of Applied Non-Classical Logics*, 11(1-2), 2001.
- [10] B. Nuseibeh, J. Kramer, and A. Finkelstein. A framework for expressing relationships between multiple views in requirements specification. *IEEE Transactions on Software Engineering*, 20(10), 1994.
- [11] B. Nuseibeh and A. Russo. Using abduction to evolve inconsistent requirements specifications. *Australian Journal of Information Systems*, 7, 1999.
- [12] M. D. Ryan. Defaults in specifications. In *Proceedings of the IEEE International Symposium on Requirements Engineering*, pages 142–149, 1993.
- [13] J. J.-P. Tsai, T. Weigert, and H.-C. Jang. A hybrid knowledge representation as a basis for requirement specification and specification analysis. *IEEE Transactions on Software Engineering*, 18(2):1076–1099.
- [14] A. van Lamsweerde, R. Darimont, and E. Leiter. Managing conflicts in goal-driven requirements engineering. *IEEE Trans. on Software Engineering*, 24(11):908–926, 1998.
- [15] V. Wiels and S. M. Easterbrook. Management of evolving specifications using category theory. In *Proceedings of the 13th International Conference on Automated Software Engineering*, 1998.