# Semantic Integration and Inconsistency

Steve Easterbrook
*Department of Computer Science, University of Toronto*
*40 St George Street, Toronto, Ontario, M5S 2E4, Canada*
*http://www.cs.toronto.edu/~sme*

**Abstract**

*The management of inconsistency between multiple viewpoints has become a central problem in the development of large software systems. In this paper we argue that the same problem occurs in the development of the semantic web, and indeed that this is the central issue in semantic integration. A common approach is to attempt to remove inconsistencies, if necessary by discarding problematic information. We argue that this approach will greatly limit the utility of the semantic web. Instead, we argue the need for formal reasoning systems that can tolerate inconsistent information. A key observation is that the problem is essentially one of model management. Rather than seeking to build a single consistent model, the challenge is to reason about the inconsistencies and dependencies between a set of inter-related partial models, and to use paraconsistent logics when reasoning with information from inconsistent ontologies.*

## 1. Viewpoint Integration in SE

For the past 15 years, we have been studying the problem of viewpoint integration in Software Engineering. Viewpoints are used in SE to support a loosely-coupled distributed approach to software development, in which different participants are able to maintain their own (partial) models of the system and its requirements, without being constrained by the need to be consistent with other participants' models [2]. By exploring the relationships between viewpoints, and the inconsistencies that arise when intended relationships do not hold, the participants discover disagreements, and understand one another's perspectives better.

The key insight of the viewpoints work is to see software development as a problem of model management, with the attendant goal of seeking coherence in information drawn from disparate sources. Software developers create models in a variety of notations to capture their current understanding of the problem and these models are rarely static. Developers analyze their models in various ways, and use the results of these analyses to improve them. They create multiple versions of their models to explore design options, and to respond to changing requirements. Hence, most of the time, design models are likely to be incomplete and inconsistent. Managing inconsistency as these models evolve is a major challenge.

In its narrowest sense, *consistency* is usually taken to mean *syntactic consistency*. In a good modeling language, syntactic consistency should correspond to the developer's intuitive notion of a "well-formed model". Hence, syntactic inconsistencies indicate simple mistakes, or slips, made by the designer. In this view, detection and resolution of inconsistency can be thought of as "model hygiene".

In our work, we have taken a much broader view of consistency. In our view, an inconsistency occurs whenever some relationship that *should* hold (of the model) has been violated. This definition has an intentional flavour: someone (e.g. the designer) *intends* that certain relationships hold. Such relationships may be internal to a model (e.g. the definition of an element should be consistent with its use), or may refer to external relationships (e.g. a model should be consistent with a particular choice of semantics, with existing standards, with good practice guidelines, or with another model, etc). This definition of inconsistency spans the semantics and pragmatics (i.e. the intended meanings and uses) of model elements, as well their syntax.

This view has several interesting consequences. Firstly, by this definition, most conceptual models are inconsistent most of the time, and attempting to remove all inconsistency is usually infeasible. Design involves finding acceptable compromises, rather than seeking perfection. Hence, in our work on consistency management, we don't view detection and removal of inconsistency as the main goal; instead, we focus on tools to explore the consistency relationships, and on reasoning techniques that tolerate inconsistency [7].

Secondly, most of the interesting consistency relationships arise implicitly as models are developed. If we wish to provide automated tools for consistency management, such consistency relationships have to be captured and represented. Thirdly, because of the intentional nature of these relationships, the set of relevant consistency relationships for a given model will change over time as the developer's intent changes.

We have made significant progress in the past 15 years in our study of these ideas.

- We have developed a number of representation schemes for capturing and managing the consistency relationships in modeling languages. These include a

first order logic for checking XML documents [6], a production rule approach for checking UML models [5] and a structural mapping technique based on graph morphisms for graphical notations [8]

- We have developed a number of reasoning techniques that tolerate inconsistency. In general, these make use of paraconsistent logics, i.e. non-classical logics whose entailment relations are not explosive under contradiction. For example, we have explored the use of a family of multi-valued logics identified by Fitting [3], and demonstrated that we can build practical reasoning engines for these logics [1].
- We have developed a theoretical framework for combining information from multiple, inconsistent sources, without first resolving the inconsistencies [8]. The composition technique we use in this framework preserves information about relative certainty and inconsistency of the source models.

## 2. Inconsistency in the Semantic Web

It now seems clear that if the semantic web is to be realized, it will not be by agreeing on a single global ontology, but rather a by weaving together a large collection of partial ontologies that are distributed across the internet [4]. We see the issues in semantic integration to be essentially the same as those in viewpoint management. In fact, the conceptual modeling tasks to which we have applied viewpoints are essentially ontology modeling tasks. For example, in requirements analysis, the models we build are domain ontologies, together with goal hierarchies and behaviour models that are based on them.

We can therefore make the following observations:

- By its very nature, the semantic web will be based on a heterogenous collection of viewpoints (partial ontologies), each constructed by a particular stakeholder for a particular purpose.
- These ontological components will not be static – they will evolve as the web services for which they were created evolve.
- For much of the time, these ontological components will be inconsistent with one another, in terms of the meanings attached to ontological elements, and the ways in which those elements are used.
- Semantic integration can only be achieved if (intentional) consistency relationships between ontological components can be captured and made explicit.
- Reasoning over the semantic web will only be possible if we have automated tools for testing these consistency relationships to identify inconsistencies.
- Fixing the inconsistencies will usually not be feasible, as this would require a globally distributed, disparate set of stakeholders to agree on and subscribe to a universal conceptual model.
- Hence, practical reasoning on the semantic web must be tolerant of inconsistency.

It should be clear by now that we believe *the* central problem in the semantic web will be managing inconsistency between ontologies. We believe our work on consistency management in the viewpoints framework suggests some promising ways forward. In particular, we believe we have practical solutions to two of the greatest challenges: representing the consistency relationships between ontologies, and reasoning over composite ontologies that contain inconsistencies. Several of the techniques described above are applicable.

We are currently investigating the application of the theoretical framework described in [8] to ontology integration. Briefly, this framework was developed for combining models in graph-based notations, where the combinations must take into account relative certainty and inconsistency of the source models. We explicitly tag elements of the models with labels indicating relative certainty and relative consistency. We call the resulting models *fuzzy viewpoints*. We then use graph morphisms to capture structural mappings between fuzzy viewpoints. Finally, we compute compositions of fuzzy viewpoints using the categorical construct of a pushout. The theoretical results on which this framework is based guarantee that we can always compute the composition, that it preserves the structure of the source models, and that no information is lost or gained in the composition. We believe that this theory provides an excellent foundation for ontology integration.

## 3. References


[1] M. Chechik, B. Devereux, S. M. Easterbrook & A. Gurfinkel "Multi-Valued Symbolic Model-Checking". To appear, IEEE Trans. on Software Engineering and Methodology, 2003.

[2] S. M. Easterbrook & B. A. Nuseibeh "Managing Inconsistencies in an Evolving Specification". 2nd IEEE Int. Symp. on Requirements Engineering (RE'95), York, UK, p48-55. Apr 1995.

[3] M. Fitting "Kleene's three-valued logics and their children". Fundamenta Informaticae, 20, 113-131, 1994

[4] J. Hendler, "Agents and the Semantic Web". IEEE Intelligent Systems, 16(2) 30--37, 2001.

[5] W. Liu, S. M. Easterbrook & J. Mylopoulos, "Rule-Based Detection of Inconsistency in UML Model". Workshop on Consistency Problems in UML-Based Software Development, 5th Int. Conference on the Unified Modeling Language, Dresden, Germany, Oct 1, 2002.

[6] C. Nentwich, W. Emmerich, A. Finkelstein and E. Ellmer, "Flexible Consistency Checking" ACM Trans. on Software Engineering and Methodology 12 (1) 28-63, 2003.

[7] B. A. Nuseibeh, S. M. Easterbrook & A. Russo, "Making Inconsistency Respectable in Software Development", J. of Systems and Software, 58 (2) 171-180. 2001.

[8] M. Sabetzadeh & S. M. Easterbrook "Analysis of Inconsistency in Graph-Based Viewpoints: A Category-Theoretic Approach". 18th IEEE Int. Conf. on Automated Software Engineering, Montreal, Oct. 6-10, 2003.