# A Foolish Consistency: Technical Challenges in Consistency Management

Anthony Finkelstein

University College London, Department of Computer Science, Gower St.,
London WC1E 6BT UK
a.finkelstein@cs.ucl.ac.uk

**Abstract.** This paper outlines the area of consistency management and argues for its importance. A motivating example is presented to support the argument. The paper sets out the key technical challenges for research in this area. A broad research agenda is outlined with some signposting of particularly interesting directions.

## 1 What is Consistency Management?

*"A foolish consistency is the hobgoblin of little minds adored by little statesmen and philosophers and divines. With consistency a great soul has simply nothing to do … speak what you think today in words as hard as cannonballs and tomorrow speak what tomorrow thinks in hard words again though it contradict everything you said today"*

**R.W. Emerson**

The everyday concept of inconsistency is easy to grasp. It is simply saying something in one place and another contradictory thing in another place. The equivalent of the standard logical notion of inconsistency in which one asserts both that *the sky is blue* and that *the sky is not blue*.

Of course inconsistency in this form is not inherently a problem until you try and base some actions in the real world upon the inconsistent assertions - selecting a coat for example. The results of doing so are varied but, depending on the veracity of the assertions, the resulting actions will interfere – simultaneously requiring wearing a raincoat and not wearing a raincoat.

Inconsistency has many causes. Foremost among these is that it results from collaboration of multiple actors, each with different opinions, views and interpretations on the real world. It is also the result of uncertainty leading to the preservation of an equivocal and hence inconsistent position. Inconsistency commonly results from errors or more rarely from deliberate falsification.

It should be clear from this that there are many circumstances in which inconsistency is acceptable, indeed desireable. In general this is in any situation where you are seeking to establish what to do prior to committing to a course of action. In such cases you may wish to have all the alternatives laid before you and as full and accurate information about the state of the world as it is possible to obtain. The consequences of this are obvious. Rather than thinking about removing inconsistency we need to think about "managing consistency". This means preserving inconsistency where it is desirable to do so, identifying inconsistency at the point where decisions are required and removing (or otherwise remedying) inconsistency prior to taking action. This requires a major change in the way we think.

Classically our concern has been to organise our information management practices so as to prevent inconsistency from arising, or using database mechanisms to remove inconsistency as close to the source as possible. Where inconsistency occurs despite our best attempts to eliminate it is viewed as an "application level" concern to be handled on a case by case basis. In a closed and relatively static corporate environment this approach might be acceptable. In the "new" information management setting of federated organisations, web publication and distributed collaborative work it patently does not work. We would like flexible and open consistency management with strategies, policies and tools defined at the generic level.

## 2 A Motivating Example

To make this discussion less abstract let us consider an example - distributed software development. Software developers, physically distributed, engage in highly collaborative work. Opinions differ and multiple inconsistent requirements and design alternatives are a particular feature of the work. Deferring commitment is an important part of the developer's armoury. The products of software development work are expressed in a variety of complex formal and semi-formal languages. These formal and semi-formal languages have more or less precise interpretations in the domain of computation.

Potential interference in this domain is reflected as a consistency relation at the level of the language itself. A typical example (for UML notations) is that an instance in a collaboration diagram for a system must be an instance of a class that appears in the class diagram for that system. Such relations can be in the form of direct mappings between the languages or by reference to a shared meta-model.

Much of the information that software developers handle is semi-structured text. The possibilities for treating consistency in this context are limited. Either we can handle consistency at the level of the structure, a formal artefact, specifying for example that the interface functional specification must contain sections with the same headings as the user manual, or at the natural language level. If at the natural language level the best we can expect are heuristic strategies.

## 3 Why is Consistency Management Difficult?

In a trivial case such as *the sky is blue*, *the sky is not blue* the inconsistency is easy to spot, if not to deal with. By contrast inconsistencies in real settings such as software development present much more challenging problems.

- The assertions can be vague or semantically ungrounded so that it is difficult to determine their precise meaning. It may be impossible to determine how the assertions relate to the real world and hence establish whether actions based on them will interfere. Assertions may be made in different (formal) languages whose relationship to each other is uncertain

- Inconsistencies can be hidden in a mass of other assertions. These assertions can be physically distributed in such a way that it is difficult to assemble the information so as to detect the inconsistency.
- An inconsistency can itself be distributed across many assertions. Consistency checking cannot always be reduced to pair-wise comparison. There may be many inconsistencies, some related and some independent, in a set of assertions. Each inconsistency can be of varying importance in the domain.
- The presence of inconsistency may pollute the set of assertions and hence prevent the use of certain technical mechanisms (such as standard first-order logic) for reasoning about the information.
- The information, which itself may be changing rapidly, may be intertwined with a complex workflow so that it is not easy to determine the points at which it is critical to establish consistency.
- In many cases inconsistencies reflect slips and minor errors or possibly delayed commitments which are relatively easy to resolve. Some inconsistencies however reflect serious conflicts with substantial knock-on consequences and may involve substantial negotiation.

## 4 What We Would Like To Do

Ideally what we would like to build a toolset which should include:

– tools to help establish, express and reason about the relationships between formal languages;
– tools to check consistency with respect to these relationships and to provide diagnostic feedback;
– where inconsistencies have been detected, tools to visualise the inconsistencies;
– tools to track inconsistencies and preserve diagnostic information in the face of ongoing change;
– tools to support resolution either through the removal of inconsistency – rectification – or by reducing the scope or severity of the inconsistency – amelioration;
– tools to support the rationale associated with consistency management.

In addition we need to be able to:

– specify policies with respect to when consistency should be checked and when resolution mechanisms should be applied;
– enforce these policies at appropriate times and in an appropriate manner.

## 5 Technical Challenges

To build such a toolset means surmounting technical challenges in a number of core areas of applied computer science.

We need an in-depth understanding of the semantics of formal languages used in particular application domains; software engineering is a good example of this. We need to be able to construct meta-models with a sound foundation and be able to reason across different formal languages expressed in terms of those meta-models. We should also be able to express constraints on the meta-model as consistency checks. In the case of an approach based on direct mappings we need "patterns" (for want of a better term) for expressing such mappings that mean the relationships are not constructed in an ad-hoc manner. More speculatively, where informal text is used we need to explore the possibility of using techniques from natural language processing to indicate potential inconsistencies.

We need to be able to check consistency in a way that is fast and highly scaleable. We can anticipate that we may have to check the consistency of very large numbers of highly complex distributed documents and other information artefacts. Consistency checking needs to be unobtrusive and low cost. This may mean devising algorithms that are computationally efficient in terms of space and performance. It may also mean devising strategies for distributing the consistency checking task and allowing the "user" to scope the consistency checks in some suitable manner. The consistency checking must be done in a way that yields useful diagnostic feedback at least localisation.

We need to devise visualisation techniques appropriate to showing consistency across complex information 'spaces'. Ideally the visualisation should also support consistency-based navigation through the information space in which it is possible to navigate across pieces of information linked by consistency relationships.

The visualisation techniques need to be linked to version and configuration control strategies that allow inconsistencies to be tracked as surrounding information changes. This is critical as we anticipate that known inconsistencies, particularly the controversial, hard to resolve ones, may be preserved as outstanding issues over relatively lengthy periods.

Conflict resolution is critical to consistency management and is both extremely difficult and little worked on, at least directly. Contributions from the fields of artificial intelligence particularly planning, computer-supported cooperative work and of course the social sciences are suggestive and provide some valuable concepts, but have yielded little which can be directly applied. Ammelioration – reducing the space, or perhaps the severity, of a conflict is an important element of resolution but demands that we have some sort of metric. This is problematic in many of the settings with which we are concerned.

It is reasonable to anticipate that users will wish to associate meta-data with inconsistencies. We have already discussed two classes of such meta-data – diagnostic and configuration information. Rationale indicating the history of discussion with respect to an inconsistency or set of inconsistencies is particularly important. Obviously this will need to be associated with the relevant information and preserved beyond the resolution of the inconsistency. The development of a rationale scheme suitable for this purpose constitutes an interesting challenge.

The specification and execution of consistency management policies provides a further technical challenge. A policy should set down what checks to perform at what point in the workflow or process to which that information relates. It may further specify what actions should be taken by way of resolution, if any, in order that the workflow or process should continue. Clearly policies themselves are domain specific but the expression of policies is a general issue. The expression of a consistency management policy is however tied to the particular way in which the workflow is described and monitored, a highly complex issue in itself.

There is not scope in this paper to present a review of related work. Probably the most recent account, though biased towards software development is [4]. The Proceedings of the Viewpoints '96 Workshop [5] and the Multi-dimensional Separation of Concerns Workshop '00 [6] reflect the growing interest among software engineers in this issue. The author has a long sequence of work addressing consistency management that includes [1], [2], [3]. Each of these papers contains a discussion of relevant contributions, these include work in AI, CSCW and distributed heterogeneous databases.

## 6 Where Now?

The end goal is the construction of a lightweight and generic set of consistency management tools which can operate across heterogeneous and distributed information, to be offered as components and web services from which an information manager will be able to assemble a consistency management solution. This goal is some way off and, as has been shown above, there are some major hurdles to surmount. Having said this, the advent of XML (eXtensible Markup Language) and associated web technologies including XML-aware databases and query languages form a very powerful substrate on which to build the sort of consistency management we envisage. The contributions of the database community to the achievement of this are eagerly sought.

## References

1. Easterbrook, S., Finkelstein, A., Kramer, J. & Nuseibeh, B.: Coordinating Distributed ViewPoints: The Anatomy of a Consistency Check, International Journal on Concurrent Engineering: Research & Applications, 2,3, (1994)
2. Emmerich, W., Finkelstein, A., Montangero, C., Antonelli, S., Armitage, S. & Stevens, R.: Managing Standards Compliance, IEEE Transactions on Software Engineering, 25, 6 (1999)
3. Finkelstein, A., Gabbay, D., Hunter, A., Kramer, J., & Nuseibeh, B.: Inconsistency Handling In Multi-Perspective Specifications, IEEE Transactions on Software Engineering, 20, 8, (1994),
4. Nuseibeh, B, Easterbrook, S. & Russo, A.: Leveraging Inconsistency in Software Development, IEEE Computer, 33, 4 (2000).
5. Viewpoints 96: An International Workshop on Multiple Perspectives in Software Development; ACM Symposium on Foundations of Software Engineering 1996 http://www.soi.city.ac.uk/~gespan/vptoc.html
6. Workshop on Multi-Dimensional Separation of Concerns in Software Engineering 2000, 22nd ICSE, http://www.research.ibm.com/hyperspace/workshops/icse2000/