Ontology Versioning on the Semantic Web

Michel Klein, Dieter Fensel
SWWS Stanford, July 30, 2001

Vrije Universiteit Amsterdam

Goal

• Stress the importance of robustness for change on the Semantic Web
• Illustrate problems caused by evolving ontologies
• Analyze requirements for versioning framework
• Propose first step towards solutions: identification and referring
Outline

- Evolving ontologies
- Ontology versioning
  - causes and consequences of ontology change
- Analysis of compatibility
- Ontology change scenario’s on the Web
  - show why we need versioning
  - extract requirements
- Proposal for identification and referring
  - discussion!

Evolving ontologies

- Info on the Web is continuously changing!  
  - however, synchronization can not be enforced
- Humans used to live with those problems:
  - heuristics and background knowledge to filter 
    outdated and wrong information
  - exponential growth reduces problems
- At the Semantic Web computers will use data!
  - invalid semantics will make all reasoning useless!
- Current SW work silently assumes stability:
  - awareness of problems, but only a few efforts
Ontology versioning

- To cope with evolving ontologies, ontology versioning is needed
  - support multiple variants of an ontology
- Methodology needed that provides:
  - methods to distinguish and recognize versions
  - procedures for changes and updates
- Definition: the ability to manage ontology changes and their effects by creating and maintaining different variants of the ontology

Causes of change

Ontology: formal specification of a shared conceptualization of a domain

- Domain change:
  - change in real world, e.g. merge of two departments
  - well known from database schema versioning
- Change in the conceptualization:
  - how the world is perceived, e.g. better understanding
  - may also be caused by adaptation for other task
- Specification change:
  - other formalism, i.e. a translation
Effects of change:

- Instance data that conforms to the ontology
- Other ontologies that are built from, or import the ontology
- Applications that use the ontology

First dependency is already complicated, others even more. We start with the first: effects of ontology changes on data that it describes

Compatibility

- Effects of changes are invalid interpretations of data
- Compatibility between ontologies and data is key issue
- Combination of different versions:
  - data with intended version of the ontology
  - data with a newer version of the ontology
  - data with an older version of the ontology
### Version combination

- **prospective use**: the use of datasources that conform to a previous version of the ontology via a newer version of the ontology.

- **retrospective use**: the use of datasources that conform to a newer version of the ontology via an older version of the ontology.

### Four types of compatibility

- **full compatible revisions**: all combinations of datasource versions and ontology versions gives a correct interpretation
  - compatible in both prospective use and retrospective use.
- **backward compatible revisions**: newer ontology version can be used to correctly interpret older data versions
  - compatible in prospective use;
- **upward compatible revisions**: newer data versions can correctly be interpreted with an older ontology version
  - compatible in retrospective use
- **incompatible revisions**: semantics of the ontology are changed in such a way that the interpretation of old datasources is affected
  - incompatible in both prospective use and retrospective use.
Remarks about compatibility

• In practice, usage can be compatible in theoretical incompatible cases!
  – when datasource only uses unaffected parts of the ontology

• Compatibility of revision is a transitive property
  – two successive compatible revisions form a new compatible revision

• However, for ontologies:
  – backward compatible can be a characteristic
    (iff all subsequent revisions are backward compatible)
  – full or upward compatible can not be a property
    (new revisions may introduce incompatibilities)

Scenario’s of change on the Web

A. The ontology is silently changed
  – the previous version is replaced by the new version without any (formal) notification

B. The ontology is visibly changed, but only the new version is accessible
  – the previous version is replaced by the new version

C. The ontology is visibly changed, and both the new version and the previous version are accessible.
Effects of scenarios

- Backward compatible change:
  - A: everything okay
  - B: nothing valid, but we could assume that the same name means the same concept
  - C: we can compare the definitions and conclude that we can still correctly interpret the data

Effects of scenarios - 2

- Incompatible change:
  - A: unexpected and unpredictable errors
  - B: nothing valid, but if we assume that the same name means the same concept, we have unpredictable errors
  - C: we can compare the definitions; if the are equal we can correctly interpret the data that is described by the part of the ontology that is not affected
Observations

- **Silent changes** often works quite well
  - backward compatible changes
  - small changes (give often only small errors)
- **Access to previous versions** is beneficial
  - compatibility can be examined, sometimes even automatically
- Agreement on meaning of “version” is needed
  - do the new definitions replace the old definitions?
  - does the same name mean the same concept?
- Incompatible changes do not necessarily make all data invalid

Requirements and approach

- Requirements for versioning methodology:
  - provide an unambiguous reference to the intended definition of concepts; (**identification**)
  - make the relation between several versions of constructs explicit; (**change tracking**)
  - provide methods to give a valid interpretation to as much data as possible (**transparent evolution**)
- Develop **opportunistic** methodology
  - aim at **maximal possible** knowledge re-use (instead of binary valid/invalid decision)
    - specify changes explicitly
    - analyze effect of changes on compatibility
  - use **minimal necessary** ontological commitment for data
Requirements for versioning

- General guideline: *exploit as much knowledge as possible*
- for every use of a concept or a relation, a versioning framework should provide an unambiguous reference to the intended definition; *(identification)*
- a versioning framework should make the relation of one version of a concept or relation to other versions of that construct explicit; *(change tracking)*
- versioning methodology on the web should make clear which part of the data can still be valid interpreted *(transparent evolution)*

Ontology identification and referring

- Proposal for ontology identification
- Starting points:
  - different *causes* of change give different *types* of change
  - there are several *types of compatibility*
  - *minimal ontological commitment* for data sources maximizes validness of data across different versions
Ontology identity

- What is the identity of an ontology?
- Ontology = specification of conceptualization
  - specification has identity
  - conceptualization has identity
- Axioms:
  - change that results in a different character representation constitutes a revision
    ⇒ new specification identity
  - change in a conceptualization
    ⇒ new ontology (conceptualization) identity
    (in case of logical equivalence: author decides whether change is only specification or also conceptualization)

Identity on the Web

- URIs? URLs? URNs?
  - Uniform Resource
    - Identifier: most general term for identification, subsumes others
    - Locator: identifier with technical hints for retrieval
    - Name: identifier that is persistent and unique
  - note: an identifier can be an instance of all three types at the same time
- Identity of ontologies:
  - Now: DAML / OIL variants, RDFS:
    - namespace mechanism used to provide identity
    - formally a URI
    - practically almost always a URL
Baseline of proposal

- Distinguish between three classes of resources:
  - files, ontologies, lines of backward compatible ontologies
- Use URL for file identification, new URI scheme for other two
- Two level numbering
  - minor numbers for backward compatible changes
  - major numbers for incompatible changes
- Concepts that only differ in minor number are equivalent
- Backward compatible additions are added as instance of “Addition” class, e.g. **Additions1.3**
- Data refers to the minimal necessary minor number

Examples

- URI for ontologies may look like:
  
  ontology://www.cs.vu.nl/~mcaklein/ontology/example/2/1/

- Line of backward compatible ontologies:
  
  ontology://www.cs.vu.nl/~mcaklein/ontology/example/2/

- Additions in version 1.3:
  - add class: **Additions1.3**
  - all other additions in version 1.3:

  ```
  NewClass
  subclass-of OldClass
  type Additions1.3
  ```
Advantages

- Not every syntactic change requires new version
- **Backward compatibility** can be exploited
  - made explicit in numbering
- Backward compatible changes can be identified
  - instances of class **Addition**
- Some **upward compatibility** is achieved
  - older ontology can be used if new additions are not necessary for new data

Remarks

- Only first step
- More detail and **validation** in practice is necessary
- Detailed analysis of **effects of changes** on data interpretation is needed
- Role of **time** should be better incorporated
- Next step: mechanism for **change specification**
  - relating different versions of concepts to each other
  - handle incompatible revisions
Summary

- **Robustness for change** is essential for the SW
- Ontology identification and referring is **first aspect** of versioning problem
- Proposed solution:
  - distinguish between file identity and ontology identity
  - make backward compatible additions explicit
  - refer to minimal necessary ontological commitment