



Reasoning with Multi-version Ontologies: Evaluation

**Zhisheng Huang, Stefan Schlobach, Frank van Harmelen, and
Michel Klein
(Vrije Universiteit Amsterdam)**

**Nuria Casellas and Pompeu Casanovas
(Universitat Autònoma de Barcelona)**

Abstract.

EU-IST Integrated Project (IP) IST-2003-506826 SEKT
Deliverable D3.5.2(WP3.5)

In this document, we present an evaluation of the Multi-version Ontology Reasoning system MORE. The framework of MORE is developed based on a temporal logic approach. We take multiple versions of the legal ontology OPJK, one of the case studies in the SEKT project, as the test data set to test the prototype of the multi-version ontology reasoning system MORE, by which we investigate and evaluate the applicability of the system.

Keyword list: ontology evaluation, ontology versioning, ontology reasoning, legal ontology

Document Id.	SEKT/2006/D3.5.2/v1.0.0
Project	SEKT EU-IST-2003-506826
Date	August 10, 2006
Distribution	public

SEKT Consortium

This document is part of a research project partially funded by the IST Programme of the Commission of the European Communities as project number IST-2003-506826.

British Telecommunications plc.

Orion 5/12, Adastral Park
Ipswich IP5 3RE
UK
Tel: +44 1473 609583, Fax: +44 1473 609832
Contact person: John Davies
E-mail: john.nj.davies@bt.com

Jozef Stefan Institute

Jamova 39
1000 Ljubljana
Slovenia
Tel: +386 1 4773 778, Fax: +386 1 4251 038
Contact person: Marko Grobelnik
E-mail: marko.grobelnik@ijs.si

University of Sheffield

Department of Computer Science
Regent Court, 211 Portobello St.
Sheffield S1 4DP
UK
Tel: +44 114 222 1891, Fax: +44 114 222 1810
Contact person: Hamish Cunningham
E-mail: hamish@dcs.shef.ac.uk

Intelligent Software Components S.A.

Pedro de Valdivia, 10
28006 Madrid
Spain
Tel: +34 913 349 797, Fax: +49 34 913 349 799
Contact person: Richard Benjamins
E-mail: rbenjamins@isoco.com

Ontoprise GmbH

Amalienbadstr. 36
76227 Karlsruhe
Germany
Tel: +49 721 50980912, Fax: +49 721 50980911
Contact person: Hans-Peter Schnurr
E-mail: schnurr@ontoprise.de

Vrije Universiteit Amsterdam (VUA)

Department of Computer Sciences
De Boelelaan 1081a
1081 HV Amsterdam
The Netherlands
Tel: +31 20 444 7731, Fax: +31 84 221 4294
Contact person: Frank van Harmelen
E-mail: frank.van.harmelen@cs.vu.nl

Empolis GmbH

Europaallee 10
67657 Kaiserslautern
Germany
Tel: +49 631 303 5540, Fax: +49 631 303 5507
Contact person: Ralph Traphöner
E-mail: ralph.traphoener@empolis.com

University of Karlsruhe, Institute AIFB

Englerstr. 28
D-76128 Karlsruhe
Germany
Tel: +49 721 608 6592, Fax: +49 721 608 6580
Contact person: York Sure
E-mail: sure@aifb.uni-karlsruhe.de

University of Innsbruck

Institute of Computer Science
Techikerstraße 13
6020 Innsbruck
Austria
Tel: +43 512 507 6475, Fax: +43 512 507 9872
Contact person: Jos de Bruijn
E-mail: jos.de-bruijn@deri.ie

Kea-pro GmbH

Tal
6464 Springen
Switzerland
Tel: +41 41 879 00, Fax: 41 41 879 00 13
Contact person: Tom Bösser
E-mail: tb@keapro.net

Sirma AI EAD, Ontotext Lab

135 Tsarigradsko Shose
Sofia 1784
Bulgaria
Tel: +359 2 9768 303, Fax: +359 2 9768 311
Contact person: Atanas Kiryakov
E-mail: naso@sirma.bg

Universitat Autònoma de Barcelona

Edifici B, Campus de la UAB
08193 Bellaterra (Cerdanyola del Vallès)
Barcelona
Spain
Tel: +34 93 581 22 35, Fax: +34 93 581 29 88
Contact person: Pompeu Casanovas Romeu
E-mail: pompeu.casanovas@uab.es

Executive Summary

In SEKT D3.5.1 we develop a framework of the multi-version ontology reasoning system MORE, based on a temporal logic approach. In this document, we present an evaluation of the system MORE. We take multiple versions of the legal ontology OPJK, one of the case studies in the SEKT project, as the test data set to test the prototype of the multi-version ontology reasoning system MORE. In this document we develop a framework to examine and analyze ontology changes and their effect on multi-version ontologies. We examine how the system MORE can support this framework of ontology versioning and its effect analysis. By the quantitative analysis of the effect, we can show various dynamic behavior of the ontologies, which provides a convenient approach for knowledge workers to obtain a better understanding on ontologies. The test indicates that the system MORE can serve as a tool for multiple versioning of ontologies.

Contents

1	Introduction	2
2	MORE: a Multi-version Ontology Reasoning System	4
2.1	Version Space and Temporal Logics	4
2.2	Functionalities of MORE	6
2.3	Ontology Comparison	6
2.4	RDF/RDFS Data Support	9
3	Legal Ontologies and Versioning	14
3.1	The SEKT legal case study	14
3.2	The OPJK versioning space	15
3.3	Analysing the OPJK versioning space	16
4	Tests, Analysis, and Evaluation	17
4.1	Ontology Versioning and Effect Space	17
4.2	Ontology Change Measure on the Version Level	18
4.2.1	Stability Measure	18
4.2.2	Difference Measure	22
4.2.3	Monotonicity Measure	25
4.3	Ontology Change Measure on the Concept Level	26
4.3.1	Stability Measure	26
4.3.2	Difference Measure	27
4.3.3	Monotonicity Measure	29
4.4	Ontology Change Measure in the Semantic Relation Level	31
4.5	Summary of the Analysis and the Findings	33
5	Discussion and Conclusions	34

Chapter 1

Introduction

Ontologies are the backbone of the Semantic Web, as they allow to share vocabulary in a semantically sound way. With the raise of the Semantic Web, the need to create ontologies has become more prominent, and even highly sensitive applications depend on ontologies, which in turn have to be of the highest possible quality. Unfortunately, building such a high-quality ontology is a very time-consuming process that often requires highly qualified professionals and domain experts over a significant time span.

Often, building an ontology can take years, and many different versions are produced (these versions are often called the *versioning space*). In this process, keeping track of modeling decisions and changes is an extremely difficult task, for the success of which a dedicated versioning system is necessary. Versioning systems are known from Software Engineering, but they are restricted to keeping track of syntactic changes. In the development of an ontology the more significant changes are often semantical. For example, adding an intermediate class to a class and its subclass will involve a syntactic change, but leaves the original subclass relation semantically intact (due to inheritance).

What can be the concrete added value of such a semantic versioning system? The most obvious use is the support of a knowledge engineer in her job of improving the ontology at hand. What use is it to introduce a relation that is already in the ontology? A versioning system might point her to the fact. More importantly, is it useful to introduce a fact, that has previously been removed from the ontology? A versioning system could point the knowledge engineer to this case, because it might suggest that a correction of the fact has previously taken place¹.

But a versioning system can also be used to analyze more general properties of the versions space. Consider a situation, where people work on the same ontology, who disagree on a particular relation between two classes. In the development process, the disputed relation will probably not remain stable, i.e. in some versions the relation will hold, in others it will not, according to who edited the latest version. Such an unstable situation can be very damaging for the overall quality of an ontology, and it is important to detect unstable relations. Part of the task of a versioning system should be to detect

¹See Chapter 4 for a detailed analysis.

instable relations or similar "problems" in the versioning space. For this purpose, we developed the system MORE, a Multi-versions Ontology Reasoning system, at the Vrije Universiteit Amsterdam as part of the European Union's SEKT project.

One of the three use cases of the SEKT project [BCB⁺04, CCLCL05, CPC⁺05b, CPC⁺05a] is a legal case study, in which an ontology is used to map questions of junior judges to a set of predefined frequently asked questions and their answers by experienced legal experts. In this context, the ontology OPJK (which stands for 'ontology for professional judicial knowledge') has been developed over the past two years. To evaluate our versioning framework we are currently recording the versioning space for the development of the OPJK. Herewith we have two aims: first, to want to evaluate the effectiveness of our versions system MORE, and secondly, we want to find epistemological properties of the versioning space.

In this document, we focus on the former, i.e. we present an evaluation of the Multi-version Ontology Reasoning system MORE. The framework of MORE is based on a temporal logic approach. We take multiple versions of the legal ontology OPJK as the test data set to test the prototype of Multi-version ontology reasoning system MORE. In this document we develop a framework to examine and analyze ontology changes and their effects on multi-versioning. We test how MORE can be used to support reasoning and management of multi-version ontologies. The tests show that the system MORE can serve as an efficient tool for multiple versioning of ontologies.

This document is organized as follows: Chapter 2 is an overview on the system MORE, and shows the new functionalities and extensions to MORE. Chapter 3 overviews the legal ontology, which serves as the case study for multi-version ontology reasoning. Chapter 4 presents the evaluation tests on MORE. Chapter 5 discusses further work, and concludes the document.

Chapter 2

MORE: a Multi-version Ontology Reasoning System

2.1 Version Space and Temporal Logics

MORE is a multi-version ontology reasoning system, which is based on a temporal logic approach[HS05a, HS05b]. Under this approach, multi-versions of an ontology are considered as a sequence of ontologies which are connected each other via change operations. Each of these ontologies has a unique name. Thus, a version space S over an ontology set Os is a set of ontology pairs, namely, $S \subseteq Os \times Os$. We use version spaces as a semantic model for our temporal logic, restricting our investigation to version spaces that present a linear sequence of ontologies. A linear version space S on an ontology set Os is denoted as a finite sequence S of ontologies as $S = (o_1, o_2, \dots, o_n)$. We use $S(i)$ to refer the i -th ontology o_i in the space. For a version space $S = (o_1, o_2, \dots, o_n)$, We call the first ontology $S(1)$ in the space the *initial version of the version space*, and the last ontology $S(n)$ the *latest version of the version space* respectively. An ordering $<$ with respect to a version space S is introduced as $o < o'$ iff o occurs prior to o' in the sequence S . We use $ontology(S)$ to denote the ontology set $Os = \{o_1, \dots, o_n\}$ of the version space S .

A temporal logic has been developed in MORE for Multi-version Reasoning[HS05a, HS05b]. The Language $\mathcal{L}+$ of the temporal logic **LTLm** is defined as an extension to the ontology language \mathcal{L} with Boolean operators and the backward temporal operators, which include the previous version operator **Prev** ϕ which denotes that the property ϕ holds in the previous version (with respect to the current version in the version space), the always-in-past operator **H** ϕ which denotes that the property ϕ always holds in any version before the current version, and the since operator ϕ **S** ψ which denoted that the property ϕ always holds (till the current version) since the property ψ holds in a version before the current version. The sometimes-in-the-past operator **P** ϕ is defined in terms of the always-in-past operator as $\neg \mathbf{H} \neg \phi$. In the temporal logic, the evaluation of a temporal formula ϕ on an ontology o (i.e., a version) in a version space S is defined as an entailment

relation[HS05a, HS05b]:

$$S, o \models \phi$$

The semantics of the temporal operators is illustrated in Figure 2.1, where arrows denote the sequence relation of ontologies in the version space, and a formula under an ontology denotes that the formula holds on the ontology. For example, the first line in the figure shows that if $\text{Prev}\phi$ holds on an ontology iff the formula ϕ hold on its previous ontology.

The temporal logic can be extended to include the future-oriented temporal operators like those in standard temporal logics[vB95, Bul70, HS91], action operators like those in dynamic logics[Har84, PT91, HTK00], and hybrid operators like those in hybrid logics[BT99, Bla00, FdRS03, AB01].

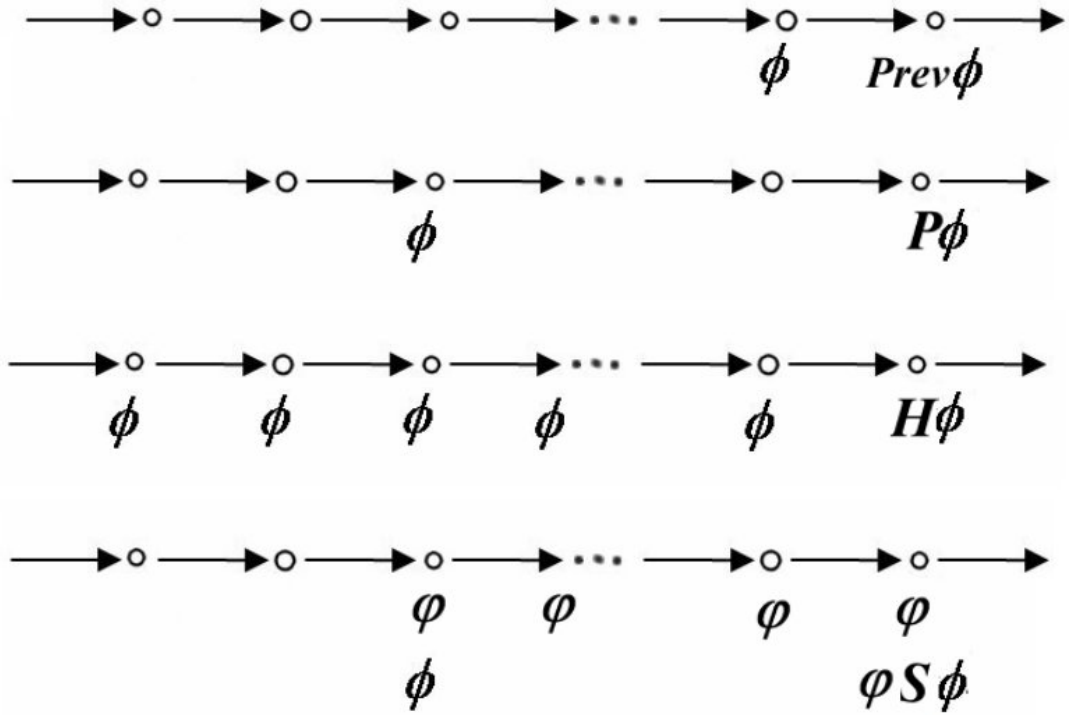


Figure 2.1: Semantics of Temporal Operators

We have implemented the prototype of MORE by using Prolog. MORE is powered by the XDIG interface [HV04], an extended DIG description logic interface for Prolog¹. MORE is designed to be a simple API for a general reasoner with multi-version ontologies. It supports extended DIG requests from other ontology applications or other ontology and metadata management systems and supports multiple ontology languages, including OWL and DIG[BMC03]². This means that MORE can be used as an interface

¹<http://wasp.cs.vu.nl/sekt/dig>

²<http://dl.kr.org/dig/>

to any description logic reasoner as it supports the functionality of the underlying reasoner by just passing requests on and provides reasoning functionalities across versions if needed. Therefore, the implementation of MORE will be independent of those particular applications or systems.

2.2 Functionalities of MORE

In SEKT Deliverable D3.5.1, we have reported that the prototype of MORE (version 1.0) have the following functionalities:

- **Temporal Reasoning Queries:** supports the temporal logic **LTLm**.
- **Ontology Comparison Queries:** supports new/obsolete/invariant concept queries with respect to children/parent/ancestor/descendant concept relations.
- **Versioning Queries :** supports version retrieval, relative version numbering, and absolute version numbering.
- **Ontology Languages:** supports the DIG format as well as the OWL language.

In this document, we would like to report the new functionalities of the prototype of MORE (version 1.5). It has the following new features:

- **Ontology Comparison Queries:** supports new/obsolete/invariant role queries with respect to the DIG role relations, like rchildren/rparent/rancestor/rdescendant, and new/obsolete/invariant instance queries.
- **Ontology Languages:** supports the RDF/RDFS data format, which are handled by SWI-Prolog's Semantic Web Library[Wie05b, Wie05a, WSW03].

2.3 Ontology Comparison

The new functionalities of MORE support the ontology comparison on the role relations and the instance relations. In Chapter 4 we are going to show that the comparison on concept/role/individual relations are very useful to examine ontology changes and their effect of multi-versioning. In the following we provide the details of the semantics of the comparison operators and its interface language in the prototype of MORE.

Many Description Logic Reasoners support so-called retrieval queries which return a set of concept names which satisfy a certain condition. For example, a children concept c' of a concept c , written $children(c, c')$, is defined as one which is subsumed by the concept c , and there exists no other named concepts between them.

Thus, the set of new/obsolete/invariant children concepts of a concept on an ontology o in the version space S is defined as follows:

$$\begin{aligned} new_{children}(S, o, c) &=_{df} \{c' | S, o \models children(c, c') \wedge \neg \mathbf{Prev} \, children(c, c')\}. \\ obsolete_{children}(S, o, c) &=_{df} \{c' | S, o \models \neg children(c, c') \wedge \mathbf{Prev} \, children(c, c')\}. \\ invariant_{children}(S, o, c) &=_{df} \{c' | S, o \models children(c, c') \wedge \mathbf{Prev} \, children(c, c')\}. \end{aligned}$$

The same definitions can be extended into the cases like parent concepts, ancestor concepts, descendant concepts. Similarly we extend those definitions on the role relations and instances relations between a concept and an individual as follows:

$$\begin{aligned} new_{rchildren}(S, o, r) &=_{df} \{r' | S, o \models rchildren(r, r') \wedge \neg \mathbf{Prev} \, rchildren(r, r')\}. \\ obsolete_{rchildren}(S, o, r) &=_{df} \{r' | S, o \models \neg rchildren(r, r') \wedge \mathbf{Prev} \, rchildren(r, r')\}. \\ invariant_{rchildren}(S, o, r) &=_{df} \{r' | S, o \models rchildren(r, r') \wedge \mathbf{Prev} \, rchildren(r, r')\}. \end{aligned}$$

where r and r' are roles, and $rchildren$ is the role children relation as defined in the DIG interface.

$$\begin{aligned} new_{instances}(S, o, c) &=_{df} \{i | S, o \models instances(c, i) \wedge \neg \mathbf{Prev} \, instances(c, i)\}. \\ obsolete_{instances}(S, o, c) &=_{df} \{i | S, o \models \neg instances(c, i) \wedge \mathbf{Prev} \, instances(c, i)\}. \\ invariant_{instances}(S, o, c) &=_{df} \{i | S, o \models instances(c, i) \wedge \mathbf{Prev} \, instances(c, i)\}. \end{aligned}$$

where c is a concept, and i is an individual.

We define the new/obsolete/invariant children concept relations of an ontology as a set of concept pairs as follows:

$$\begin{aligned} new_{children}(S, o) &=_{df} \{\langle c, c' \rangle | S, o \models children(c, c') \wedge \neg \mathbf{Prev} \, children(c, c')\}. \\ obsolete_{children}(S, o) &=_{df} \{\langle c, c' \rangle | S, o \models \neg children(c, c') \wedge \mathbf{Prev} \, children(c, c')\}. \\ invariant_{children}(S, o) &=_{df} \{\langle c, c' \rangle | S, o \models children(c, c') \wedge \mathbf{Prev} \, children(c, c')\}. \end{aligned}$$

Similar definitions can be extended into new/obsolete/invariant role relations with respect to $rchildren/rparents/rancestors/rddescendants$ relations and new/obsolete/invariant instance relations between a concept and an individual, like these:

$$new_{rchildren}(S, o) =_{df} \{\langle r, r' \rangle | S, o \models rchildren(r, r') \wedge \neg \mathbf{Prev} \, rchildren(r, r')\}.$$

$obsolete_{rchildren}(S, o) =_{df} \{ \langle r, r' \rangle | S, o \models \neg rchildren(r, r') \wedge \mathbf{Prev} rchildren(r, r') \}.$
 $invariant_{rchildren}(S, o) =_{df} \{ \langle r, r' \rangle | S, o \models rchildren(r, r') \wedge \mathbf{Prev} rchildren(r, r') \}.$
 where r and r' are roles, and $rchildren$ is the role children relation as defined in the DIG interface.

$new_{instances}(S, o) =_{df} \{ \langle c, i \rangle | S, o \models instances(c, i) \wedge \neg \mathbf{Prev} instances(c, i) \}.$
 $obsolete_{instances}(S, o) =_{df} \{ \langle c, i \rangle | S, o \models \neg instances(c, i) \wedge \mathbf{Prev} instances(c, i) \}.$
 $invariant_{instances}(S, o) =_{df} \{ \langle c, i \rangle | S, o \models instances(c, i) \wedge \mathbf{Prev} instances(c, i) \}.$

Those query supports are sufficient to evaluate the consequences of the ontology changes and the differences among multi-version ontologies. We will discuss them in Chapter 4.

In the following, we will introduce the interface language for those comparison queries in MORE. Queries on ontology comparison like those on new/obsolete/invariant concept/role relations are expressed as the corresponding XML-encoded statements. Here is an example of queries on what are new children role relations with respect to the role 'hasPet' on the ontology 'MadCow2', compared with the ontology 'MadCow1':

```

<?xml version="1.0" encoding="ISO-8859-1" ?>
<askm xmlns="http://wasp.cs.vu.nl/sekt/more/lang"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://wasp.cs.vu.nl/sekt/more/more.xsd">
<querym id="new child roles of the role hasPet on MadCow2 compared MadCow1"
versionSpace="MadCow" ontology="MadCow2">
<newOnRole role="hasPet" type="rchildren"
comparedOntology="MadCow1"/>
</querym>
</askm>

```

If the query does not contain a specific ontology to make comparisons with, the ontology will be compared to with the previous one (in the linear sequence of the version space).

```

<querym id="new child roles of the role hasPet on MadCow2 on versionSpace"
versionSpace="MadCow" ontology="MadCow2">
<newOnConcept role="bird" type="rchildren"/>
</querym>

```

The compared roles can be specified as a list in the query as shown below:

```

<querym id="new child roles for a role list on MasCow2"
versionSpace="MadCow" ontology="MadCow2">
<newOnRole type="rchildren">
<ratom name="hasPet"/>
<ratom name="hasColor"/>
</newOnRole>
</querym>

```

If there is no particular role stated, that means the query should be carried on all roles in the ontology:

```
<querym id="new child roles on MadCow2"
  versionSpace="MadCow"  ontology="MadCow2">
  <newOnRole type="rchildren"/>
</querym>
```

If there is no stated type in the query, that means the query should be carried on all role relations, namely, on rchildren/rparents/rancestors/rdescendants role relations.

```
<querym id="new child roles on MadCow2"
  versionSpace="MadCow"  ontology="MadCow2">
<newOnRole/>
</querym>
```

Obsolete/Invariant role relations can be stated in queries similarly:

```
<querym id="obsolete child roles on MadCow2"
  versionSpace="MadCow"  ontology="MadCow2">
<obsoleteOnRole/>
</querym>
<querym id="invariant child roles on MadCow2"
  versionSpace="MadCow"  ontology="MadCow2">
<invariantOnRole/>
</querym>
```

The interface language for role relations and individual relations are summarized in Figure 2.2 and Figure 2.3 respectively. The interface language for concept relation is summarized in Figure 2.4, which is also available in SEKT Deliverable D3.5.1.

2.4 RDF/RDFS Data Support

In the MORE interface, the TELL language is designed to be a natural extension to the TELL requests in the DIG interface for multi-version ontology reasoning. However, note that the extension to the DIG interface would not hinder its independence of any particular reasoning language. In MORE, a TELL request contains a tellm (i.e., the tell request in MORE) element in its body. MORE supports various ontology data formats, like DIG, OWL and RDF/RDFS. The ontology data in DIG and OWL are converted into its DIG data format as the XDIG's internal data format. Reasoning on those data is supported by an external DIG reasoner, like RACER[HM01], FACT++[Hor99]. In order to achieve more efficient way to manage RDF/RDFS data, the new prototype of MORE (version 1.5) uses SWI-Prolog's Semantic Web Library to manage and reason on RDF/RDFS data. Thus, if ontology data are specified as a RDF/RDFS data format in the TELL language, they need no an external DL reasoner for its reasoning Support.

New role	<querym id=ID versionSpace=S ontology=o> <newOnRole role=r type=Type/> </querym>	$new_{Type}(S, o, r)$
Obsolete role	<querym id=ID versionSpace=S ontology=o> <obsoleteOnRole role=r type=Type/> </querym>	$obsolete_{Type}(S, o, r)$
Invariant role	<querym id=ID versionSpace=S ontology=o> <invariantnRole role=r type=Type/> </querym>	$obsolete_{Type}(S, o, r)$
New role on all roles	<querym id=ID versionSpace=S ontology=o> <newOnRole type=Type/> </querym>	$new_{Type}(S, o)$
Obsolete role on all roles	<querym id=ID versionSpace=S ontology=o> <obsoleteOnRole type=Type/> </querym>	$obsolete_{Type}(S, o)$
Invariant role on all roles	<querym id=ID versionSpace=S ontology=o> <invariantOnRole type=Type/> </querym>	$invariant_{Type}(S, o)$
New role without a type	<querym id=ID versionSpace=S ontology=o> <invariantOnRole/> </querym>	$new_{type}(S, o)$ for all <i>type</i>
Obsolete role without a type	<querym id=ID versionSpace=S ontology=o> <obsoleteOnRole/> </querym>	$obsolete_{type}(S, o)$ for all <i>type</i>
Invariant role without a type	<querym id=ID versionSpace=S ontology=o> <invariantOnRole/> </querym>	$invariant_{type}(S, o)$ for all <i>type</i>
New role compared with arbitrary ontology	<querym id=ID versionSpace=S ontology=o> <newOnRole role=c type=Type comparedOntology=o' /> </querym>	New <i>type</i> role of <i>r</i> in <i>o</i> , compared with <i>o'</i>
New role relation for a role list	<querym id=ID versionSpace=S ontology=o> <newOnRole type=Type > <ratom name = r_1 > ... <catom name = r_n > < /newOnRole> </querym>	$new_{Type}(S, o, r)$ for all $r \in \{r_1, \dots, r_n\}$

where *type* = rchildren/rparents/rancestors/rdescendants

Figure 2.2: Query patterns on role comparison

New individual	<querym id=ID versionSpace=S ontology=o> <newOnIndividual concept=c/> </querym>	$new_{instances}(S, o, c)$
Obsolete individual	<querym id=ID versionSpace=S ontology=o> <obsoleteOnIndividual concept=c/> </querym>	$obsolete_{instances}(S, o, c)$
Invariant individual	<querym id=ID versionSpace=S ontology=o> <invariantOnIndividual concept=c/> </querym>	$obsolete_{instances}(S, o, c)$
New individual on all concepts	<querym id=ID versionSpace=S ontology=o> <newOnIndividual/> </querym>	$new_{instances}(S, o)$
Obsolete individual on all concepts	<querym id=ID versionSpace=S ontology=o> <obsoleteOnIndividual/> </querym>	$obsolete_{instances}(S, o)$
Invariant individual on all concepts	<querym id=ID versionSpace=S ontology=o> <invariantOnIndividual/> </querym>	$invariant_{instances}(S, o)$
New individual compared with arbitrary ontology	<querym id=ID versionSpace=S ontology=o> <newOnIndividual concept=c comparedOntology=o' /> </querym>	new instances of c in o , compared with o'
New individual relation for a concept list	<querym id=ID versionSpace=S ontology=o> <newOnIndividual> <catom name = c_1 > ... <catom name = c_n > < /newOnIndividual> </querym>	$new_{instances}(S, o, c)$ for all $c \in$ $\{c_1, \dots, c_n\}$

Figure 2.3: Query patterns on individual comparison

New concept	<querym id=ID versionSpace=S ontology=o> <newOnConcept concept=c type=Type/> </querym>	$new_{Type}(S, o, c)$
Obsolete concept	<querym id=ID versionSpace=S ontology=o> <obsoleteOnConcept concept=c type=Type/> </querym>	$obsolete_{Type}(S, o, c)$
Invariant concept	<querym id=ID versionSpace=S ontology=o> <invariantOnConcept concept=c type=Type/> </querym>	$obsolete_{Type}(S, o, c)$
New concept on all concepts	<querym id=ID versionSpace=S ontology=o> <newOnConcept type=Type/> </querym>	$new_{Type}(S, o)$
Obsolete concept on all concepts	<querym id=ID versionSpace=S ontology=o> <obsoleteOnConcept type=Type/> </querym>	$obsolete_{Type}(S, o)$
Invariant concept on all concepts	<querym id=ID versionSpace=S ontology=o> <invariantOnConcept type=Type/> </querym>	$invariant_{Type}(S, o)$
New concept without a type	<querym id=ID versionSpace=S ontology=o> <invariantOnConcept/> </querym>	$new_{type}(S, o)$ for all $type$
Obsolete concept without a type	<querym id=ID versionSpace=S ontology=o> <obsoleteOnConcept/> </querym>	$obsolete_{type}(S, o)$ for all $type$
Invariant concept without a type	<querym id=ID versionSpace=S ontology=o> <invariantOnConcept/> </querym>	$invariant_{type}(S, o)$ for all $type$
New concept compared with arbitrary ontology	<querym id=ID versionSpace=S ontology=o> <newOnConcept concept=c type=Type comparedOntology=o' /> </querym>	New $type$ concept of c in o , compared with o'
New concept relation for a concept list	<querym id=ID versionSpace=S ontology=o> <newOnConcept type=Type > <catom name = c_1 > ... <catom name = c_n > < /newOnConcept> </querym>	$new_{Type}(S, o, c)$ for all $c \in \{c_1, \dots, c_n\}$

where $type = \text{children/parents/ancestors/descendants}$

Figure 2.4: Query patterns on concept comparison

The Semantic Web Library of SWI-Prolog have been proved to be very efficient for reading, querying, and storing semantic web documents. The library can handle upto about 2 million RDF triples on an ordinary computer (256MB memory, Pentium 1.5Ghz).

Ontology data which will be handled by SWI-Prolog's Semantic Web Library have to be told by a tellm statement which refers to their URLs like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<tellm xmlns="http://wasp.cs.vu.nl/sekt/more/lang"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://wasp.cs.vu.nl/sekt/more/more.xsd">
  <ontology name="ontoRDF" language="rdf"
    url="file:ontology/legalcase/ontoRDF.rdf"/>
  <ontology name="ontoRDF" language="rdfs"
    url="file:ontology/legalcase/ontoRDF.rdfs"/>
  <ontology name="ontoRDF2dig" language="dig"
    url="http://wasp.cs.vu.nl/ontology/legalcase/ontoRDF2.dig2.xml"/>
  <ontology name="ontoRDF3" language="rdf"
    url="file:ontology/legalcase/ontoRDF3.rdf"/>
  <ontology name="ontoRDF3" language="rdfs"
    url="file:ontology/legalcase/ontoRDF3.rdfs"/>
  <ontology name="oplk" language="rdf"
    url="file:ontology/legalcase/oplk.rdf"/>
  <ontology name="oplk" language="rdfs"
    url="file:ontology/legalcase/oplk.rdfs"/>
  <ontology name="oplkowl" language="owl"
    url="file:ontology/legalcase/oplk.owl"/>
  <ontology name="opjk" language="rdf"
    url="file:ontology/legalcase/opjk.rdf"/>
  <ontology name="opjk" language="rdfs"
    url="file:ontology/legalcase/opjk.rdfs"/>
  <ontology name="opjkowl" language="owl"
    url="file:ontology/legalcase/opjk.owl"/>

  <versionSpace name="legalcase1">
  <pair ontology1="ontoRDF" ontology2="ontoRDF2dig"/>
  <pair ontology1="ontoRDF2dig" ontology2="ontoRDF3"/>
  <pair ontology1="ontoRDF3" ontology2="oplk"/>
  <pair ontology1="oplk" ontology2="oplkowl"/>
  <pair ontology1="oplkowl" ontology2="oplk"/>
  <pair ontology1="opjk" ontology2="opjkowl"/>
  </versionSpace>
</tellm>
```

In this example, some of ontologies, like ontoRDF, ontoRDF3, are specified as ones in RDF/RDFS data format. Their management and reasoning are supported by SWI-Prolog's Semantic Web Machinery. That is done by MORE automatically. Users need not to care about how SWI-Prolog's Semantic Web Library is used. Some of ontologies, like ontoRDF2dig and oplkowl, are claimed to be ones in DIG/OWL format. They rely on an external DL reasoner which supports the DIG interface for their reasoning support.

Chapter 3

Legal Ontologies and Versioning

The aim of SEKT is to develop and exploit semantically-based knowledge technologies in order to support document management, content management, and knowledge management in knowledge intensive workplaces. Specifically, SEKT aims at designing appropriate utilities to users in three main areas: digital libraries, the engineering industry, and the legal domain, providing them with quick access to the right pieces of information at the right time. In this chapter, we report on the ontologies of the legal case study of the SEKT project and their versioning. We will first briefly introduce the legal case study, before describing the relevant version space for the OPJK ontology in more detail.

3.1 The SEKT legal case study

More details regarding the legal case study can be found in SEKT deliverable D10.2.1[CPC⁺05b].

In the legal case study, the accomplished tasks so far provide both the quantitative and qualitative data necessary to assess both the context of users, (newly recruited Spanish judges), and their specific needs with regard to the technology under development. In particular, these data give an insight on institutional, organizational, and individual constraints that could either facilitate or block the introduction of SEKT technologies in judicial units. From this set of data, the Ontology of Professional Judicial Knowledge is built.

Until now, the legal ontologies have been built up with several purposes: information retrieval, statute retrieval, normative linking, knowledge management or legal reasoning. Although the legal domain remains very sensitive to the features of regional or national statutes and regulations, some of the Legal-Core Ontologies (LCO) are intended to share a common kernel of legal notions. However, LCO remain in the domain of a general knowledge shared by legal theorists, national or international jurists and comparative lawyers. This use case study fed on the idea of the existence of a Professional Legal Knowledge (PLK) which is: (i) shared among members of the judicial professional

group (e.g. judges, attorneys, prosecutors...); (ii) learned and conveyed formally or most often informally in specific settings (e.g. the Judicial School, professional associations -the Bar, the Judiciary...-); (iii) expressible through a mixture of natural and technical language (legalese, legal slang); (iv) non-equally distributed among the professional group; (v) non-homogeneous (elaborated on individual bases); (vi) universally comprehensible by the members of the profession (there is a sort of implicit identification principle).

The data (nearly 800 competency questions) gathered from judicial interviews by the legal case study, allows the construction of an ontology based on professional judicial knowledge. Knowledge derived from the daily practice at courts. The Ontology of Professional Judicial Knowledge (OPJK) developed by the legal case study team has been learnt from scratch out of nearly 400 competency questions and has, currently, nearly 90 concepts, 100 relations and 70 individuals. The following top domain concepts have been identified: *acto_procesal*, *organo_judicial*, *calificacion_juridica*, *documento_procesal*, *fase_procesal*, *jurisdiccion*, *proceso_judicial*, *profesion_juridica*, *rol_procesal*, *rol_familiar* and *sancion*. In this deliverable, we will use the ontology OPJK as the test data to evaluate the system of ontology versioning and management.

The Legal Case Study Prototype has been designed taking into account two main considerations: (i) an accurate searching system, with advanced technology, that goes beyond traditional searching algorithms, capable of reliable search over a vast FAQ repository; (ii) a design that supports a fast, usable, modular, extensible, scalable, improving implementation. The first point might be achieved by using some techniques like ontologies to model legal case domains and NLP techniques. The second point is achieved by leveraging on some software technology patterns, like a multistage searching cycle for successive approach to FAQ pair target or pluggable searching stage engines. The final design is flexible, modular, scalable, customizable and suitable for the prototype.

3.2 The OPJK versioning space

The OPJK ontology lies at the core of the legal case study within SEKT, as it links natural language questions (formulated by newly recruited Spanish judges) with a repository of frequently asked questions (FAQs) with their corresponding answers provided by more experienced judges. To this aim, OPJK contains definitions for the most relevant judicial terms in the professional knowledge of law professionals with respect to this set of FAQs. As this is highly specialised knowledge producing a new version of the OPJK is a difficult task. In practice, a team of legal experts meets in regular intervals to decide on the relevant changes to the OPJK where the decision is taken w.r.t. the questions in the FAQs.

The OPJK versioning space has been built to reflect on this structured way of dealing with the frequently asked questions, and in principle, it is a linear space with usually one new version per FAQ. Not only is this a direct mirror of the creation process, it also constitutes an opportunity to analyse the epistemic process in a systematic way.

Technically, we collected 27 versions of the OPJK up to now, and many more will be expected in the near future.

3.3 Analysing the OPJK versioning space

We can now analyse the OPJK versioning space with two different motivations: first, as a well-constructed (and documented) versioning space of a complex ontology, and secondly, as a formalisation of an epistemological process, in which knowledge elicitation is made explicit in ontological changes.

In this document we will only address the first issue, and leave the latter to the following SEKT deliverable 3.5.3. More concretely, we will analyse the OPJK versioning space from two perspective. First, we will study measures on the temporal dimension of the version space, i.e. search whether we can detect peculiarities in the process at a whole. Is the process a stable, monotonic one, where information is added to new versions? Or is there constantly information retracted, suggesting a more ad hoc process? In the following Chapter we will try to answer these kinds of questions in a systematic way.

Chapter 4

Tests, Analysis, and Evaluation

4.1 Ontology Versioning and Effect Space

When an ontology is changed, knowledge engineers may want to know the ramification of the change. They may want to know whether or not the change may lead to some unintended effect on the ontology. Usually those effect have to be judged from their semantic implications, which cannot be judged by a syntactic checking on the ontology. Moreover, knowledge engineers may want to have some qualitative or quantitative evaluation on the changes they have done. For example, they may want to know whether or not those changes are big or small, in the sense that they lead to big or small semantic difference from its previous version. They may want to know whether or not those changes are monotonic or non-monotonic in the sense that they would not obsolete any implied semantic relation.

In order to measure ontology changes and their effect, we have to check the ramification of an ontology change on all possible semantic relations on the concept/role/individual relations of an ontology. We call the set of all possible semantic relations the *Effect Space* of a version space. By the notion of the effect space, we will develop a framework to examine and evaluate ontology versioning and their changes. We want to propose a formal approach to measure those kinds of changes and their effect, so that we can evaluate ontology changes with respect to their qualitative/quantitative properties. All of the ontology changes can be examined under an effect space which covers all of the possible changes and their ramification on the semantic relation with respect to concepts/roles/individuals. In this document, we consider only effect spaces which are characterized by the new/obsolete/invariant relation on concepts/roles/individuals on a version space. Namely, an effect space consists of those relations are new/obsolete/invariant concept relations with respect to children/parents/ancestors/descendants relations, new/obsolete/invariant role relations with respect to rchildren/rparents/ancestors/descendant relations, and new/obsolete/invariant instance relation between concepts and individuals.

Suppose that an ontology o in a version space S consists of about n_c concepts, n_r roles,

and n_i individuals. The number of possible concept relations on new/obsolete/invariant with respect to children/parents/ancestors/descendants aspects is $N_C(S, o) = n_c \times n_c \times 3 \times 4$. The number of possible instance relations between a concept and an individual is $N_I(S, o) = n_c \times n_i \times 3$. The number of possible role relations is $N_R(S, o) = n_r \times n_r \times 3 \times 4$. Therefore, the effect space has $N(S, o) = N_C(S, o) + N_R(S, o) + N_I(S, o)$ possible semantic relations at each version. Suppose that version space S consists of n_v version ontologies and each ontology has almost the same numbers of concept/role/individual, then the number of possible semantic relations in a version space is $N(S, o) \times n_v$.

In the following we will measure the ontology change and their effects from the following different levels:

- **Version level:** We examine ontology changes on a single version, so that their effects can be displayed in a timeline of a version space, from which we provide a dynamic view on ontology changes.
- **Concept/Role level:** We examine ontology changes on a single concept or role to see its dynamic properties, from which we can obtain the picture of concepts/roles for their stability, difference, and the monotonicity.
- **Semantic Relation Level:** We examine single semantic relation to see its temporal aspect in a version space.

4.2 Ontology Change Measure on the Version Level

In this section, we measure ontology changes and their effect on the version level, so that the difference on changes can be presented as a timeline on a version space. Moreover, we measure ontology changes with respect to the following criteria respectively:

- **stability:** how stable among the semantic relations when an ontology is subjected to a change which leads to a new version.
- **difference:** what are differences, more exactly what are new, among semantic relations when an ontology has been changed.
- **monotonicity:** whether or not the existence of some semantic relations which hold in the previous version do not hold in the current version?

The relation among those three properties are illustrated in Figure 4.1.

4.2.1 Stability Measure

We measure the stability of an ontology as the sets of its invariant concept/role/individual relations which are compared with its previous version. Thus, we have the following formal definitions:

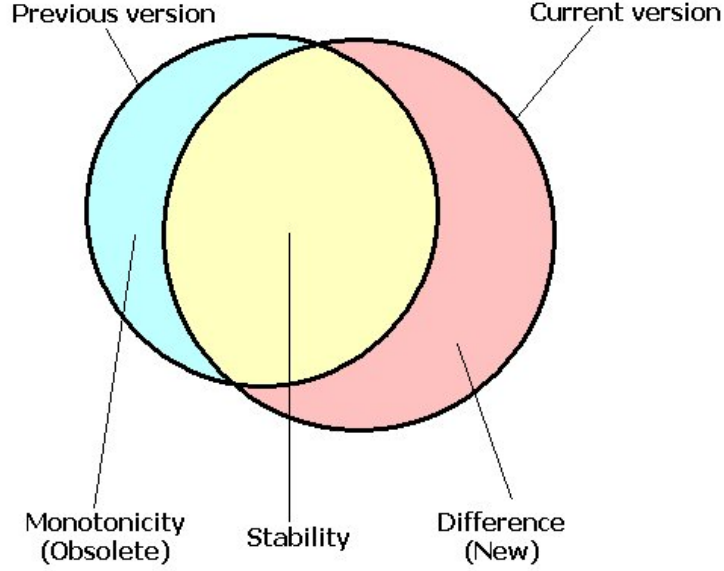


Figure 4.1: The relation among the stability, the difference, and the monotonicity.

Definition 4.2.1 (Concept Stability)

$$InvariantonConcept(S, o) = \{ \langle type, c, c' \rangle \mid \langle c, c' \rangle \in invariant_{type}(S, o) \text{ and } type \in \{children, parents, ancestors, descendants\} \}$$

Definition 4.2.2 (Role Stability)

$$InvariantonRole(S, o) = \{ \langle type, r, r' \rangle \mid \langle r, r' \rangle \in invariant_{type}(S, o) \text{ and } type \in \{rchildren, rparents, rancestors, rdescendants\} \}$$

Definition 4.2.3 (Individual Stability)

$$InvariantonIndividual(S, o) = \{ \langle type, c, i \rangle \mid \langle c, i \rangle \in invariant_{instances}(S, o) \}$$

Definition 4.2.4 (Ontology Stability)

$$Invariant(S, o) = InvariantonConcept(S, o) \cup InvariantonRole(S, o) \cup InvariantonIndividual(S, o).$$

Figure 4.2 shows the timeline differences of the cardinality of the stability sets in the version space OPJK¹. A way to normalize the stability measure so that the normalized

¹So far the OPJK ontology has flat role relations. Therefore, we do not count any role relation of the OPJK ontology in this document.

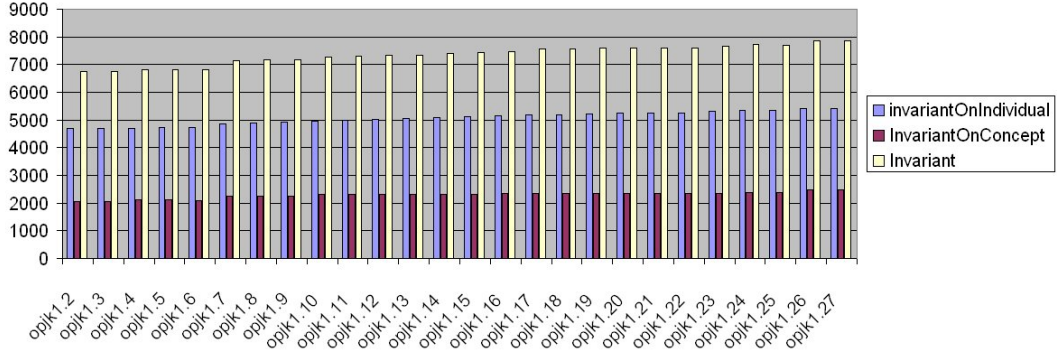


Figure 4.2: Timeline of the stability of the OPJK ontologies (opjk1.1-opjk1.27)

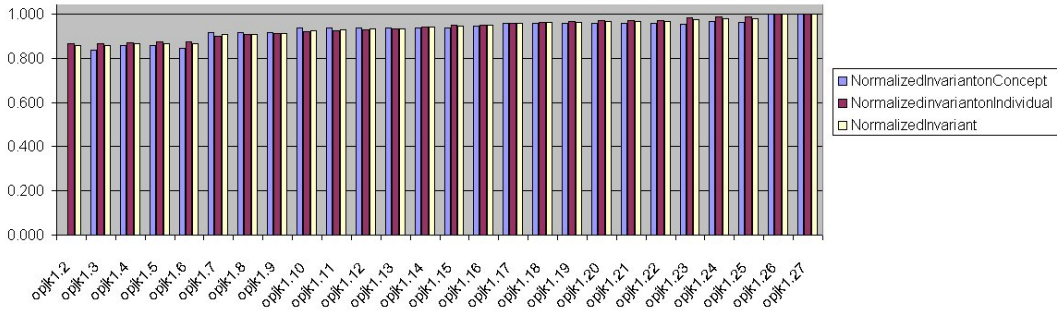


Figure 4.3: Timeline of the normalized stability of the OPJK ontologies (opjk1.1-opjk1.27)

values fall into the set $[0, 1]$ is to divide the cardinality by the maximal cardinality in the set, as defined in Definition 4.2.5. The normalized stability is shown in Figure 4.3. This kind of normalization gives the impression that the ontology is getting more and more stable with time goes.

Definition 4.2.5 (Normalized Stabilities)

$$\begin{aligned}
 NIoC(S, o) &= |IoC(S, o)| / \max(\{|IoC(S, o')| : o' \in \text{ontology}(S)\}) \\
 NIoR(S, o) &= |IoR(S, o)| / \max(\{|IoR(S, o')| : o' \in \text{ontology}(S)\}) \\
 NIoI(S, o) &= |IoI(S, o)| / \max(\{|IoI(S, o')| : o' \in \text{ontology}(S)\}) \\
 NI(S, o) &= |Invariant(S, o)| / \max(\{|Invariant(S, o')| : o' \in \text{ontology}(S)\})
 \end{aligned}$$

where $NIoC$ = Normalized InvariantonConcept, $NIoR$ = Normalized InvariantonRole, $NIoI$ = Normalized InvariantonIndividual, NI = Normalized Invariant, IoC = InvariantonConcept, IoR =InvariantonRole, and IoI =InvariantonIndividual.

A more reasonable approach to normalize the stability is to divide them by the corresponding relation numbers, i.e. N_C, N_R, N_I , and N , in a single version o of the effect

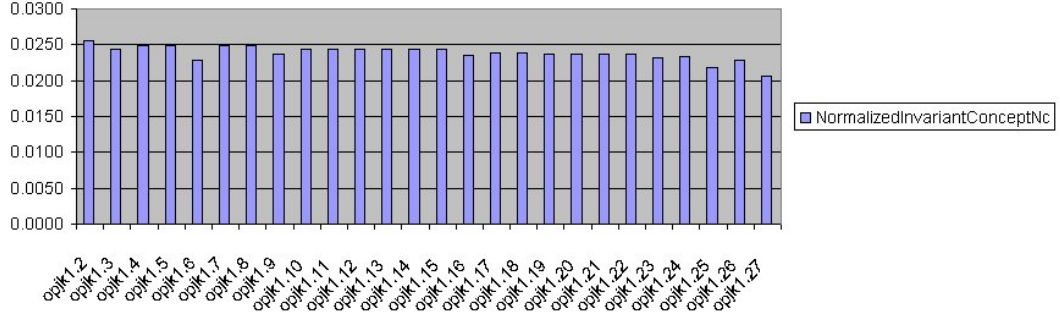


Figure 4.4: Timeline of the normalized concept stability of the OPJK ontologies w.r.t. the effect space (opjk1.1-opjk1.27)

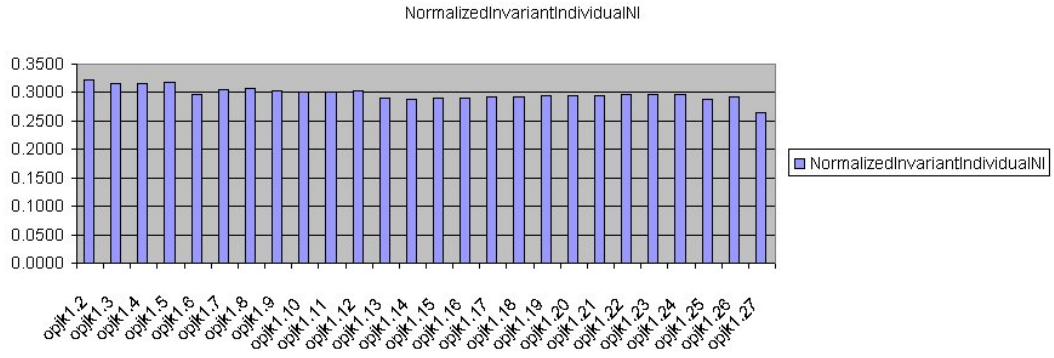


Figure 4.5: Timeline of the normalized individual stability of the OPJK ontologies w.r.t. the effect space (opjk1.1-opjk1.27)

space in a version space S , which are defined in Section 4.1. Namely, we normalize the stability with respect to its effect space.

Definition 4.2.6 (Normalized Stability with respect to Effect Space)

$$\begin{aligned}
 NIoCNC(S, o) &= |IoC(S, o)|/N_C(S, o) \\
 NIoRNR(S, o) &= |IoR(S, o)|/N_R(S, o) \\
 NIoINI(S, o) &= |IoI(S, o)|/N_I(S, o) \\
 NIN(S, o) &= |Invariant(S, o)|/N(S, o)
 \end{aligned}$$

where $NIoCNC$ = Normalized InvariantonConcept w.r.t. N_C , $NIoRNR$ = Normalized InvariantonRole w.r.t. N_R , $NIoINI$ = Normalized InvariantonIndividual w.r.t. N_I , NIN = Normalized Invariant w.r.t. N , IoC = InvariantonConcept, IoR =InvariantonRole, and IoI =InvariantonIndividual.

The timelines of the normalized statbility with respect to concept/individual/general relation in the effect spaces are shown in Figure 4.4, Figure 4.5, and Figure 4.6 respectively. From the timeline graphs, we can see that there exist some difference of the stability with time goes. Actually the last few versions (opjk1.25 and opjk1.27) are less stable

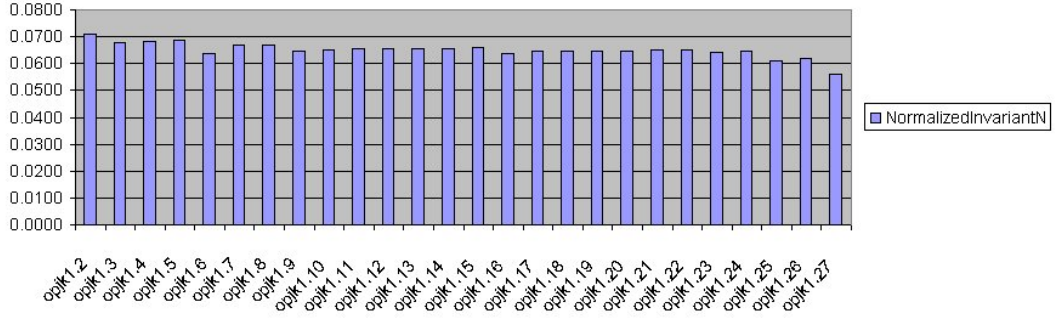


Figure 4.6: Timeline of the normalized stability of the OPJK ontologies w.r.t. the effect space (opjk1.1-opjk1.27)

than their previous versions. In the version opjk1.25 and opjk1.27, new subclasses are created and some classes have been changed their positions in the concept hierarchy.

4.2.2 Difference Measure

The new/obsolete concept/role/individual relations show the difference of an ontology from its previous version. They can be used to measure how big a change has been done on the ontology. We are particularly interested in the difference measure by new concept/role/ individual relations. Similar with those definition on invariance, we define the difference measure on the new concept/role/individual relations. Thus, the new on an ontology and its normalized one are defined as follows:

Definition 4.2.7 (New on Ontology)

$$New(S, o) = NewonConcept(S, o) \cup NewonRole(S, o) \cup NewonIndividual(S, o).$$

Definition 4.2.8 (Normalized New)

$$\begin{aligned} NNoC(S, o) &= |NoC(S, o)| / \max(\{|NoC(S, o')| : o' \in ontology(S)\}) \\ NNoR(S, o) &= |NoR(S, o)| / \max(\{|NoR(S, o')| : o' \in ontology(S)\}) \\ NNoI(S, o) &= |NoI(S, o)| / \max(\{|NoI(S, o')| : o' \in ontology(S)\}) \\ NN(S, o) &= |New(S, o)| / \max(\{|New(S, o')| : o' \in ontology(S)\}) \end{aligned}$$

where $NNoC$ = Normalized NewonConcept, $NNoR$ = Normalized NewonRole, $NNoI$ = Normalized NewonIndividual, NN = Normalized New relation, NoC = NewonConcept, NoR = NewonRole, and NoI = NewonIndividual.

Figure 4.7 is the timeline of the new relation of the OPJK ontology which shows the amount of different concept/individual relations of all ontologies in the version space

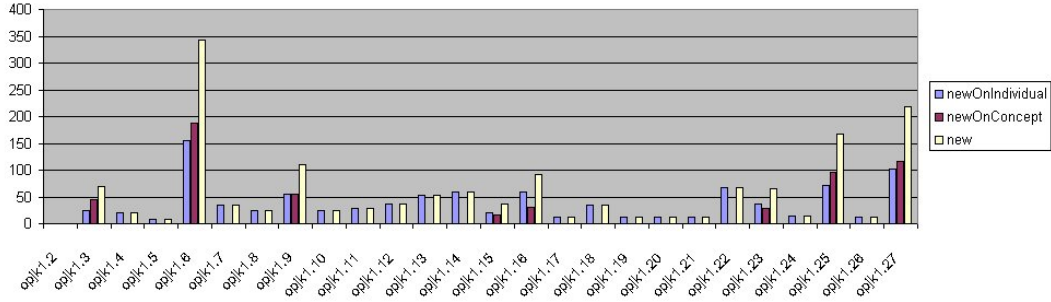


Figure 4.7: Timeline of the new relation of the OPJK ontologies

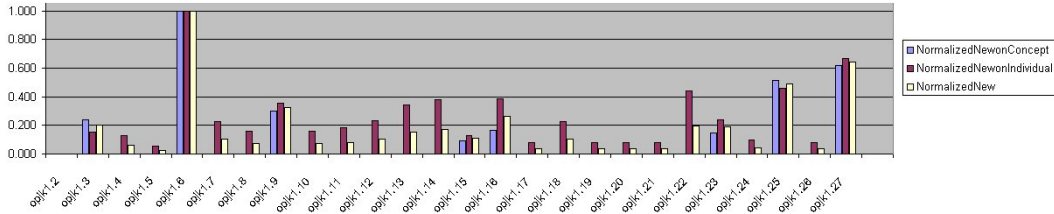


Figure 4.8: Timeline of the normalized new relation of the OPJK ontologies

OPJK. The normalized new relation is shown in Figure 4.8. From the timeline figures we know that most of the changes on the OPJK ontology are small. The change on `opjk1.5`, which is shown on new relation on `opjk1.6`, is significantly big. The biggest change occurs on `opjk1.6`, the second biggest change occurs on `opjk1.27`, and the third one occurs on `opjk1.25`.

Definition 4.2.9 (Normalized New with respect to Effect Space)

$$\begin{aligned}
 NNoCNC(S, o) &= |NoC(S, o)| / N_C(S, o) \\
 NNoRNR(S, o) &= |NoR(S, o)| / N_R(S, o) \\
 NNoINI(S, o) &= |NoI(S, o)| / N_I(S, o) \\
 NNN(S, o) &= |New(S, o)| / N(S, o)
 \end{aligned}$$

where $NNoCNC$ = Normalized NewonConcept w.r.t. N_C , $NNoRNR$ = Normalized NewonRole w.r.t. N_R , $NNoINI$ = Normalized NewonIndividual w.r.t. N_I , NNN = Normalized New relation w.r.t. N , NoC = NewonConcept, NoR = NewonRole, and NoI = NewonIndividual.

Similarly we introduce the normalized difference measure with respect to the effect space. Figure 4.9 is the timeline of the normalized new relation of the OPJK ontology with respect to the effect space. The result shows a similar difference measure as that of normalization with respect to the maximal number. Namely, the biggest change occurs on `opjk1.6`, the second biggest change occurs on `opjk1.27`, and the third biggest change

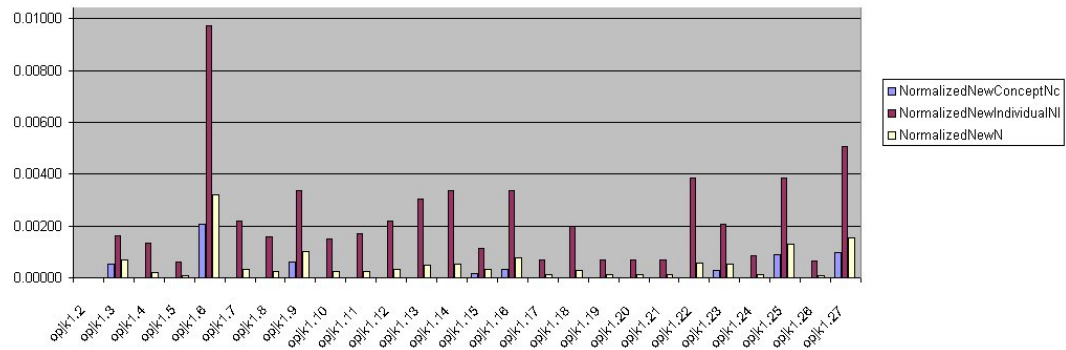


Figure 4.9: Timeline of the normalized new of the OPJK ontologies w.r.t. the effect space (opjk1.1-opjk1.27)

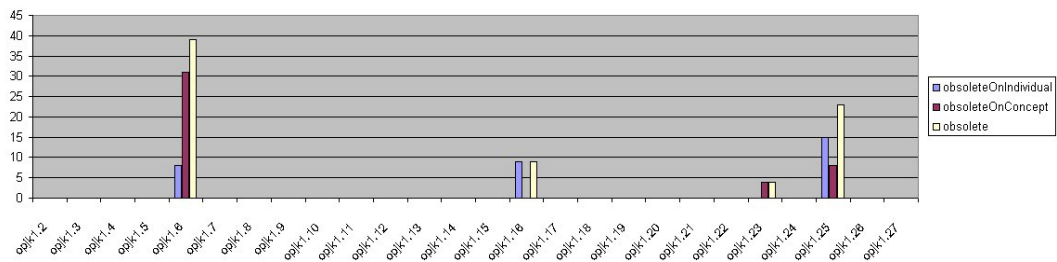


Figure 4.10: Timeline of the obsolete relation of the OPJK ontologies

occurs on opjk1.25. The two different normalization approaches tell a similar result under this OPJK versioning scenario.

4.2.3 Monotonicity Measure

The obsolete concept/role/individual relations show that some semantic relations on an ontology which hold in the previous version do not hold in the current version any more. Therefore, it can be considered as a kind of measure for the monotonicity/non-monotonicity of an ontology change. By the monotonicity we mean that the change does not obsolete any previously held property, otherwise it is called a non-monotonic change.

Therefore, we have the following formal definitions:

Definition 4.2.10 (Ontology Monotonicity)

$$\begin{aligned} \text{Obsolete}(S, o) = & \text{ObsoleteonConcept}(S, o) \cup \\ & \text{ObsoleteonRole}(S, o) \cup \\ & \text{ObsoleteonIndividual}(S, o). \end{aligned}$$

Similarly the normalized obsolete is measured by

Definition 4.2.11 (Normalized obsolete)

$$\begin{aligned} \text{NOoC}(S, o) &= |\text{OoC}(S, o)| / \max(\{|\text{OoC}(S, o')| : o' \in \text{ontology}(S)\}) \\ \text{NOoR}(S, o) &= |\text{OoR}(S, o)| / \max(\{|\text{OoR}(S, o')| : o' \in \text{ontology}(S)\}) \\ \text{NOoI}(S, o) &= |\text{OoI}(S, o)| / \max(\{|\text{OoI}(S, o')| : o' \in \text{ontology}(S)\}) \\ \text{NO}(S, o) &= |\text{Obsolete}(S, o)| / \max(\{|\text{Obsolete}(S, o')| : o' \in \text{ontology}(S)\}) \end{aligned}$$

where NOoC = Normalized ObsoleteonConcept, NOoR = Normalized ObsoleteonRole, NOoI = Normalized ObsoleteonIndividual, NO = Normalized Obsolete relation, OoC = ObsoleteonConcept, OoR = ObsoleteonRole, and OoI = ObsoleteonIndividual.

Figure 4.7 is the timeline of the new relation of the OPJK ontology which shows the amount of different concept/individual relations of all ontologies in the version space OPJK. A finding from this measure is that adding new individuals occurs more often than adding new concepts. This suggests that most additions were at the instance level, i.e., at A-box, not at T-box. However, once a new concept is added, it usually leads to a bigger change than that of a new individual. The normalized new relation is shown in Figure 4.8. From the timeline figures we know that most of the changes on the OPJK ontology are small. The change on opjk1.5, which is shown on new relation on opjk1.6, is significantly big. Figure 4.10 is the timeline of the obsolete relation of the OPJK ontology, which shows that only opjk1.6, opjk1.23, and opjk1.25 have some obsolete concept relations and obsolete individual relations, and opjk1.16 has only some obsolete individual relations.

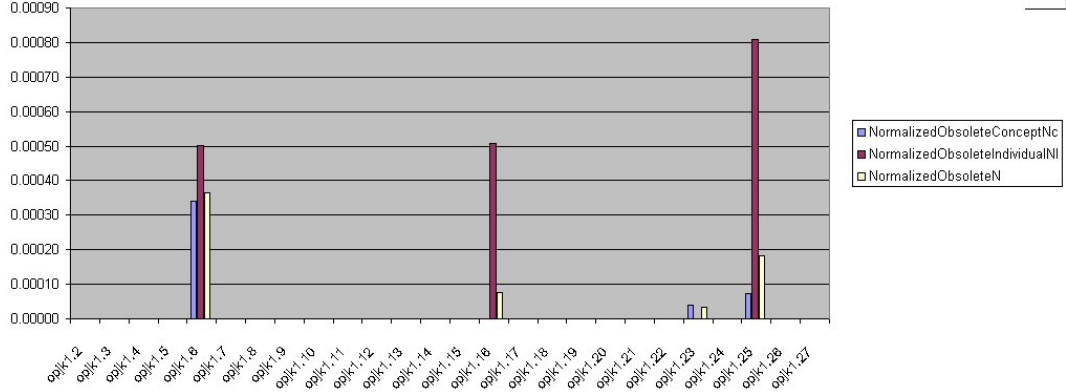


Figure 4.11: Timeline of the normalized obsolete of the OPJK ontologies w.r.t. the effect space (opjk1.1-opjk1.27)

Definition 4.2.12 (Normalized obsolete with respect to Effect Space)

$$\begin{aligned}
 NOoCNC(S, o) &= |OoC(S, o)| / N_C(S, o) \\
 NOoRNR(S, o) &= |OoR(S, o)| / N_R(S, o) \\
 NOoINI(S, o) &= |OoI(S, o)| / N_I(S, o) \\
 NON(S, o) &= |Obsolete(S, o)| / N(S, o)
 \end{aligned}$$

where $NOoCNC$ = Normalized ObsoleteonConcept w.r.t. N_C , $NOoRNR$ = Normalized ObsoleteonRole w.r.t. N_R , $NOoINI$ = Normalized ObsoleteonIndividual w.r.t. N_I , NON = Normalized Obsolete relation w.r.t. N , OoC = ObsoleteonConcept, OoR = ObsoleteonRole, and OoI = ObsoleteonIndividual.

Figure 4.11 is the timelines of the normalized obsolete relation of the OPJK ontology with respect to effect space. Those timelines show that opjk1.25 has the biggest nonmonotonicity effect and opjk1.23 has the second biggest nonmonotonicity effect with respect to the individual relation.

4.3 Ontology Change Measure on the Concept Level

In this section we measure ontology changes and their effects on individual concepts, so that we can obtain the message which concept/role in the ontology is more stable than others, by which we can find the core of an ontology.

4.3.1 Stability Measure

We measure the stability of a concept in a version space as the amount of its invariant relations which are compared with its previous version. Thus, we have the following formal definitions:

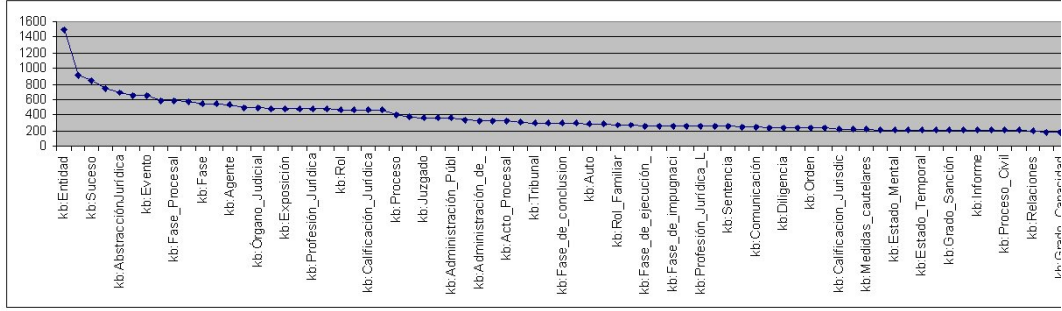


Figure 4.12: The concept stability of the OPJK ontologies (opjk1.1-opjk1.15)

Definition 4.3.1 (Concept Stability of a Concept)

$$\text{ConceptInvariant}(S, c) = \{ \langle \text{type}, c, c' \rangle \mid \langle c, c' \rangle \in \text{invariant}_{\text{type}}(S, o) \text{ and } \text{type} \in \{\text{children}, \text{parents}, \text{ancestors}, \text{descendants}\} \text{ and } o \in \text{ontology}(S) \}$$

Definition 4.3.2 (Role Stability of a Role)

$$\text{RoleInvariant}(S, r) = \{ \langle \text{type}, r, r' \rangle \mid \langle r, r' \rangle \in \text{invariant}_{\text{type}}(S, o) \text{ and } \text{type} \in \{r\text{children}, r\text{parents}, r\text{ancestors}, r\text{descendants}\} \text{ and } o \in \text{ontology}(S) \}$$

Definition 4.3.3 (Individual Stability of a Concept)

$$\text{IndividualInvariant}(S, c) = \{ \langle \text{type}, c, i \rangle \mid \langle c, i \rangle \in \text{invariant}_{\text{instances}}(S, o) \text{ and } o \in \text{ontology}(S) \}$$

Definition 4.3.4 (Concept Stability)

$$\text{ConceptStability}(S, c) = \text{ConceptInvariant}(S, c) \cup \text{IndividualInvariant}(S, c).$$

Figure 4.12 shows the stability of the concepts in the OPJK ontology. Similarly we normalize the concept stability measure by dividing the invariance cardinality by the maximal invariance cardinality. The normalized concept stability is shown in Figure 4.13. The individual stability is shown in Figure 4.14. Figure 4.15 shows top 30 most stable concepts from opjk1.1 till opjk1.15 in the OPJK ontology.

4.3.2 Difference Measure

Similar with those in the difference measure in the ontology level, we measure the difference on the concept level in terms of the new concept relations and new instances relations.

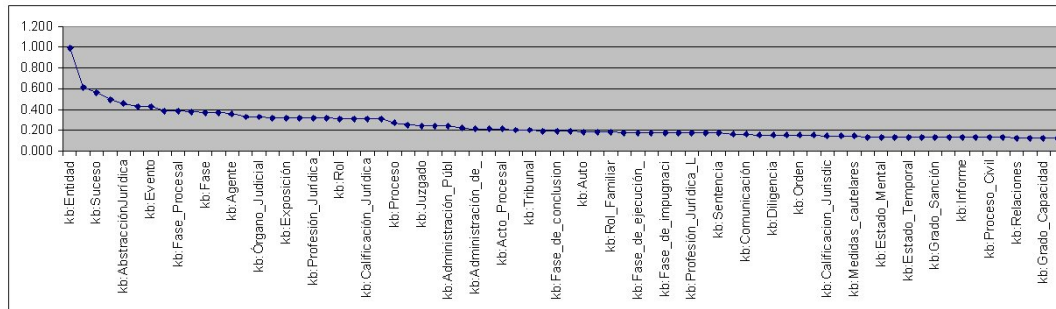


Figure 4.13: The normalized concept stability of the OPJK ontologies (opjk1.1-opjk1.15)

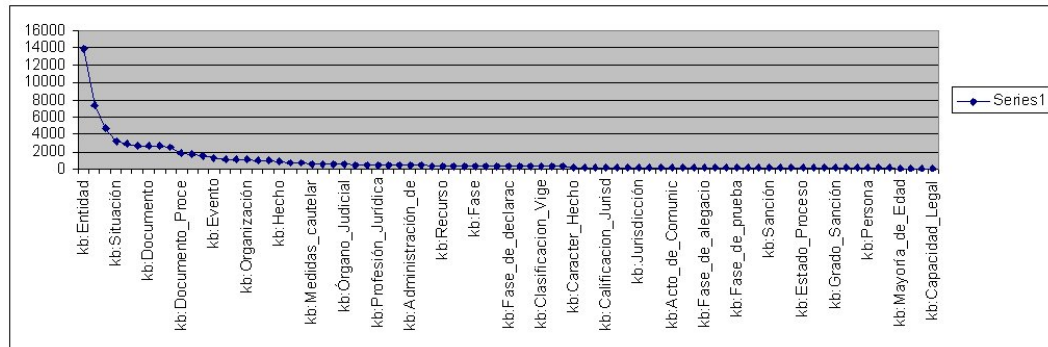


Figure 4.14: The individual stability of the OPJK ontologies (opjk1.1-opjk1.15)

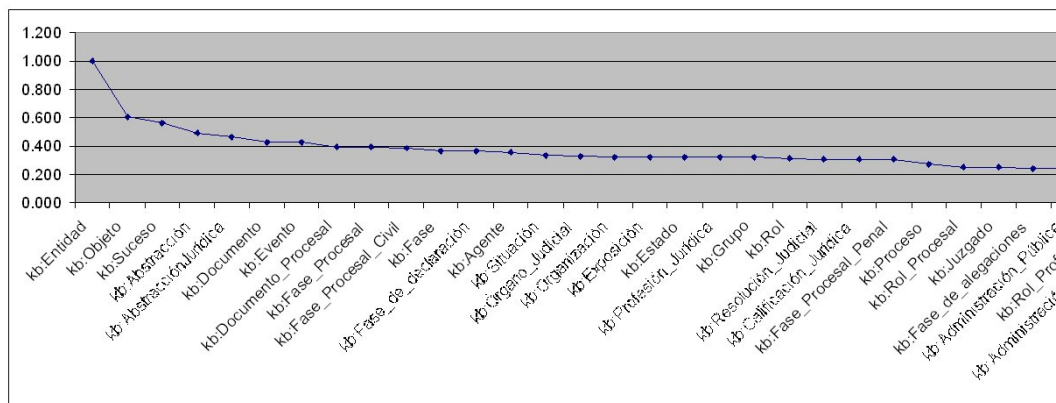


Figure 4.15: Top 30 most stable concepts in the OPJK ontologies

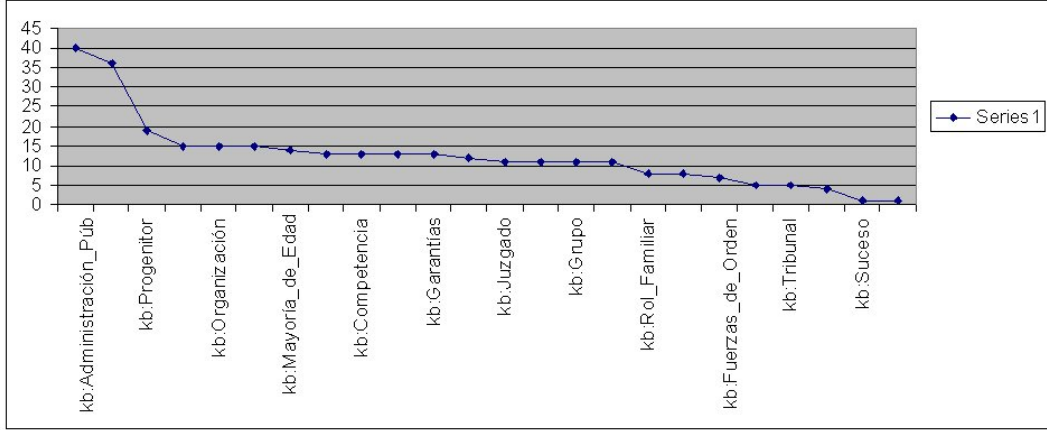


Figure 4.16: The concept difference of the OPJK ontologies (opjk1.1-opjk1.15)

Definition 4.3.5 (Concept Difference of a Concept)

$$ConceptNew(S, c) = \{ \langle type, c, c' \rangle \mid \langle c, c' \rangle \in new_{type}(S, o) \text{ and } type \in \{children, parents, ancestors, descendants\} \text{ and } o \in ontology(S) \}$$

Definition 4.3.6 (Individual Difference of a Concept)

$$IndividualNew(S, c) = \{ \langle type, c, i \rangle \mid \langle c, i \rangle \in new_{instances}(S, o) \text{ and } o \in ontology(S) \}$$

Definition 4.3.7 (Concept Difference)

$$ConceptDifference(S, c) = ConceptDifference(S, c) \cup IndividualDifference(S, c).$$

Figure 4.16 shows the concept difference of concepts in the OPJK ontology. The normalized concept difference is shown in Figure 4.17. The individual difference is shown in Figure 4.18. Figure 4.19 shows top 20 mostly effected concepts from opjk1.1 till opjk1.15² in the OPJK ontology.

4.3.3 Monotonicity Measure

Again, we measure the monotonicity in the concept level by using the obsolete concept/role/individual relations.

²This kind of the analysis data were obtained before version opjk1.16 was created. In this document we provide only the data about opjk1.1 - opjk1.15. The data about other versions will be available at the website of MORE: <http://wasp.cs.vu.nl/sekt/more>.

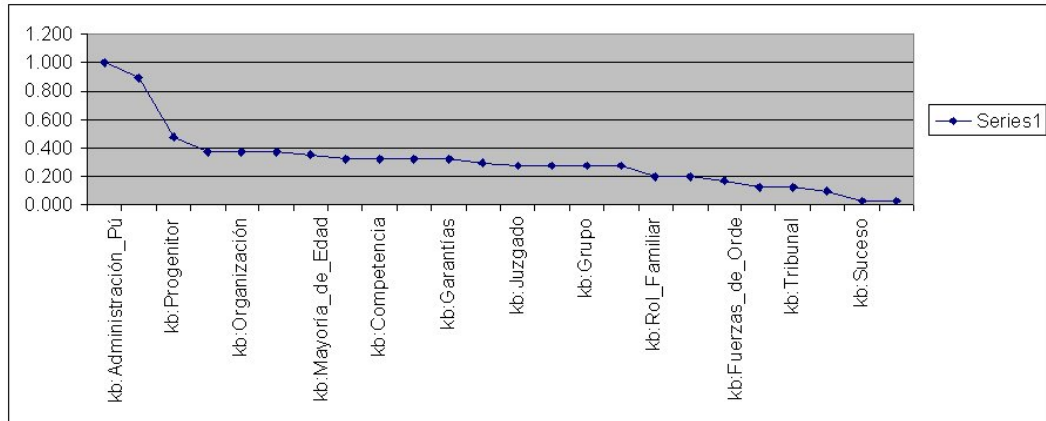


Figure 4.17: The normalized concept difference of the OPJK ontologies (opjk1.1-opjk1.15)

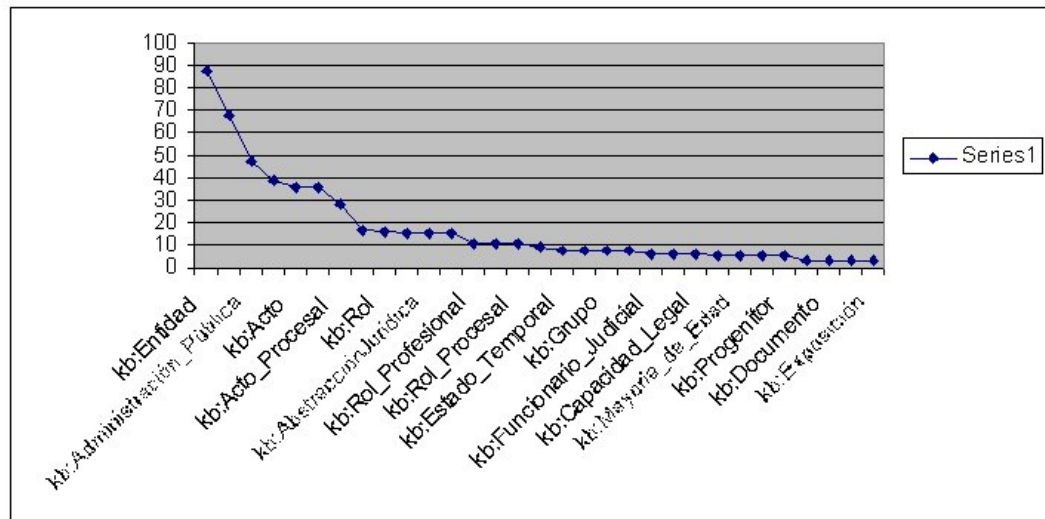


Figure 4.18: The individual difference of the OPJK ontologies (opjk1.1-opjk1.15)

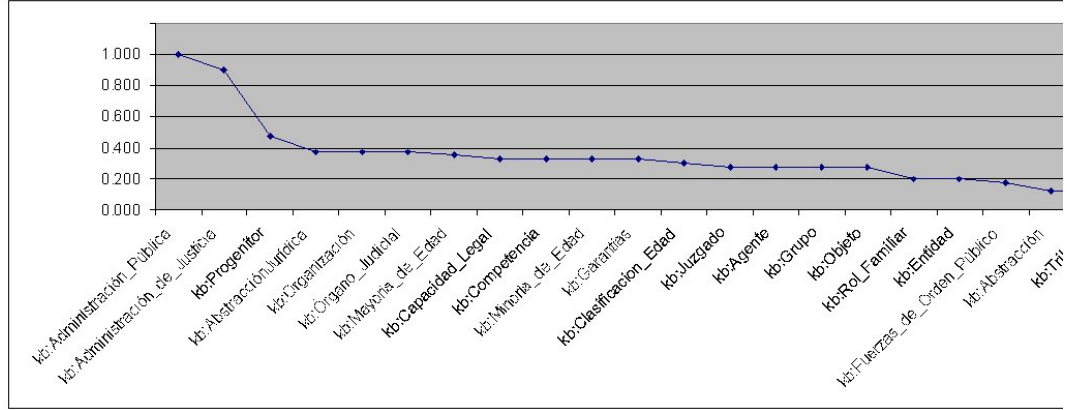


Figure 4.19: Top 20 mostly effected concepts in the OPJK ontologies (opjk1.1-opjk1.15)

Definition 4.3.8 (Concept Obsolete of a Concept)

$$\text{ConceptObsolete}(S, c) = \{ \langle \text{type}, c, c' \rangle \mid \langle c, c' \rangle \in \text{obsolete}_{\text{type}}(S, o) \text{ and } \text{type} \in \{\text{children}, \text{parents}, \text{ancestors}, \text{descendants}\} \text{ and } o \in \text{ontology}(S) \}$$

Definition 4.3.9 (Individual Obsolete of a Concept)

$$\text{IndividualObsolete}(S, c) = \{ \langle \text{type}, c, i \rangle \mid \langle c, i \rangle \in \text{obsolete}_{\text{instances}}(S, o) \text{ and } o \in \text{ontology}(S) \}$$

Definition 4.3.10 (Concept Monotonicity)

$$\text{ConceptMonotonicity}(S, c) = \text{ConceptObsolete}(S, c) \cup \text{IndividualObsolete}(S, c).$$

As shown in Figure 4.10, the OPJK ontology has some obsolete concept relations only at opjk1.6, opjk1.23, and opjk1.25. It is less interesting to show them as the figures. The details of the obsolete relations can be obtained from the website of MORE³.

4.4 Ontology Change Measure in the Semantic Relation Level

A semantic relation can be examined with respect to its temporal aspects. If a property on the ontology is changed once, it is never changed back again in any sequel version as illustrated in Figure 4.20. We consider that kind of change as a stable one, because it occurs only once in the whole version space. The corresponding temporal query of stable change on a property can be expressed as

$$\neg\phi \text{SH} \phi.$$

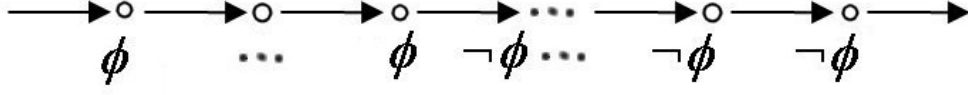
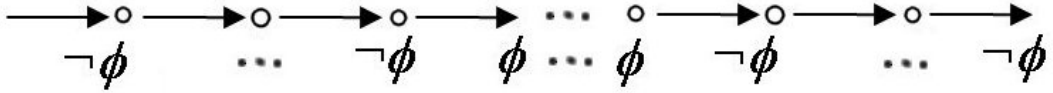


Figure 4.20: Stable change

The query on the existence of only two changes with respect to a property ϕ , as shown in Figure 4.21, can be expressed as

$$\neg\phi\mathbf{SPrev}(\phi\mathbf{SH}\neg\phi).$$

Figure 4.21: Only two changes with respect to ϕ

Define the only- n -times-change operator as follows:

$$\mathit{Change}(1, \phi) =_{df} \neg\phi\mathbf{SH}\phi.$$

$$\mathit{Change}(n, \phi) =_{df} \neg\phi\mathbf{SPrev}\mathit{Change}(n - 1, \neg\phi),$$

for $n > 1$.

We examine the OPJK ontology for its changes with respect to the temporal aspects. It shows that all concept relations on the OPJK ontology from opjk1.1 till opjk1.27 are stable. That can be confirmed by examining the obsolete concept relations only in opjk1.6, opjk1.23, and opjk1.25, because the timeline presentation of the obsolete concept in Figure 4.10 which shows that obsolete concept relations occur only in those versions. We will extend our evaluation to more complex temporal queries in our next research.

³<http://wasp.cs.vu.nl/sekt/more>

4.5 Summary of the Analysis and the Findings

In the following we summarize the main findings and observations of the OPJK ontology versioning from the analysis and the evaluation, by which we want to obtain a better understanding on the design of the OPJK. Moreover, we will discuss the implications for the development process of ontologies.

- In general the ontology is getting more stable with time goes, from Figure 4.2 it is observable that the level of stability increases with time. During the first versions, important decisions on concept and sub-concept level are made.
- The results provided above in this deliverable, also show that some recent OPJK versions are less stable than their previous ones (Figure 4.4). The property indicates that some conceptual restructuring occurs on OPJK. In these last 3 versions, due to insight provided by the data (competency questions) the modellers modified modelling decisions made in previous versions; modify subclass relationships and add children to certain classes. For example, in version 1.25, a new subclass of *FamilyRole* is created and instances belonging to another class have been moved to it. Also a new subclass is added in version 1.27 and existing instances are re-assigned to it.
- From the results it can also be affirmed that adding new individuals occurs more often than adding new concepts. This suggests that most additions were at the instance level, i.e., at A-box, not at T-box. However, when a new concept is added, it usually leads to a bigger change than that by a new individual.
- Finally, the data also shows that so far most of the changes on OPJK are small. Few semantic relations are obsolete. That means that the current methodology which is introduced in the design of the OPJK ontology is reliable.

Regarding the implications for ontology development, ontology modellers may rely on the ontology stability measurement to be able to analyze the modelling decisions that involved a great level of instability. In the same way, if the amount of instability inflicted by adding or modifying subclasses could be assessed, different modelling options could be sought to promote stability. From the modelling point of view, it is interesting to see how certain knowledge experts discussions are mirrored in the stability measures of the ontology.

Measurements on the concept level might also be very useful for modellers. Regarding OPJK, the list of the most stable concepts in OPJK does not include *Acto* and *Hecho*, that, although they are in the ontology since the initial versions, they do not appear within the most 30 stable concepts; these two concepts have been part of an ongoing discussion within the modelling team, as it can be learnt from the concept stability measurement. Concept stability measurements are not only useful to know which concepts are not stable, but also to learn that some concepts (such as *Objeto* in OPJK), have been stable against all odds.

Chapter 5

Discussion and Conclusions

In this document we presented first evaluation results of the MORE versioning system on the OPJK versioning space.

The OPJK is an ontology which was developed within the SEKT project's legal case study to support the semantic classification of questions of junior legal experts to a set of Frequently Asked Questions. The OPJK has been developed by legal experts over a significant time, and a number of successive versions have been recorded in the so-called versioning space. This space contains valuable information on the knowledge elicitation process, but is also an interesting test case for the use of semantic versioning techniques in the creation of ontologies.

The MORE tool was developed in Amsterdam within SEKT, and described in detail in SEKT deliverable 3.5.1. It offers semantic versioning support for ontology development based on a combination of change detection and Linear Temporal Logic.

The experiments we conducted were three-fold. First, we studied the properties of the overall versioning space by checking for stability, novelty and monotonicity of the process. More concretely, we studied at which time semantic relations between classes were added, deleted or remained the same. This indicated that there are different types of changes in the versioning space, the smaller ones with only minor, often cosmetic, variations, and the more substantial ones, in which many new semantic relations change. The second study was a more detailed analysis on concept level. Here, we identified the most stable concepts in the OPJK ontologies, those with the most frequent changes, and the most commonly effected concepts. Finally, we provided an initial case study on the stability of change in the versioning space using the power of the temporal logic which underlies MORE.

Extending our evaluation to more complex temporal queries is one of the next steps in our research. Although the current research has shown the potential of the mechanism and the practicability of the tool support, a more detailed analysis of the temporal properties of the versioning space seems promising, but also needed to evaluate the suitability of our versioning language for the purpose at hand.

Finally, a more philosophical discussion has now become possible, and will be in our focus in the near future: a detailed analysis of the knowledge elicitation process through analysis of the version space. Here, in close co-operation with partners from the different working packages within SEKT, we hope to extend our understanding of the role of particular legal terms or concepts in the light of the application, and their function within the semantic structure.

Bibliography

- [AB01] C. Areces and P. Blackburn. Bringing them all together. *Journal of Logic and Computation*, 11(5):657–669, 2001. Special Issue on Hybrid Logics. Areces, C. and Blackburn, P. (eds.).
- [BCB⁺04] V.R. Benjamins, J. Contreras, M. Blázquez, L. Rodrigo, P. Casanovas, and M. Poblet. The sekt use legal case components: ontology and architecture. In T.B. Gordon, editor, *Legal Knowledge and Information Systems*, pages 69–77. IOS Press, Amsterdam, 2004.
- [Bla00] P. Blackburn. Representation, reasoning, and relational structures: a hybrid logic manifesto. *Logic Journal of the IGPL*, 8(3):339–365, 2000.
- [BMC03] S. Bechhofer, R. Möller, and P. Crowther. The DIG description logic interface. In *International Workshop on Description Logics (DL2003)*. Rome, September 2003.
- [BT99] P. Blackburn and M. Tzakova. Hybrid languages and temporal logic. *Logic Journal of the IGPL*, 7(1):27–54, 1999.
- [Bul70] R. Bull. An approach to tense logic. *Theoria*, 36:282–300, 1970.
- [CCLCL05] V.R. Benjamins P. Casanovas, J. Contreras, J. M. López-Cobo, and L. Lemus. Iuriservice: An intelligent frequently asked questions system to assist newly appointed judges. In V.R. Benjamins et al, editor, *Law and the Semantic Web*, pages 205–522. Springer-Verlag, London, Berlin, 2005.
- [CPC⁺05a] P. Casanovas, M. Poblet, N. Casellas, J. Contreras, R. Benjamins, and M. Blázquez. Supporting newly-appointed judges: A legal knowledge management case study. *Journal of Knowledge Management*, 2005.
- [CPC⁺05b] P. Casanovas, M. Poblet, N. Casellas, J-J. Vallbé, F. Ramos, V.R. Benjamins, M. Blázquez, J. Contreras, and J. Gorronogoitia. Legal scenario. case study intelligent integrated decision support for legal professionals. Project Report Report D10.2.1, SEKT, 2005.
- [FdRS03] M. Franceschet, M. de Rijke, and B. H. Schlingloff. Hybrid logics on linear structures: expressivity and complexity. In *Proceedings TIME 2003*, pages 166–173. IEEE Computer Society Press, 2003.

- [Har84] D. Harel. Dynamic logic. In D. Gabbay and F. Guenther, editors, *Handbook of Philosophical Logic Volume II — Extensions of Classical Logic*, pages 497–604. D. Reidel Publishing Company: Dordrecht, The Netherlands, 1984.
- [HM01] V. Haarslev and R. Möller. Description of the racer system and its applications. In *Proceedings of the International Workshop on Description Logics (DL-2001)*, pages 132–141. Stanford, USA, August 2001.
- [Hor99] I. Horrocks. Fact and ifact. In *Proceedings of the International Workshop on Description Logics (DL’99)*, pages 133–135, 1999.
- [HS91] J. Y. Halpern and Y. Shoham. A propositional modal logic of time intervals. *Journal of the ACM*, 38(4):935–962, October 1991.
- [HS05a] Z. Huang and H. Stuckenschmidt. Reasoning with multiversion ontologies. Project Report Report D3.5.1, SEKT, 2005.
- [HS05b] Z. Huang and H. Stuckenschmidt. Reasoning with multiversion ontologies: a temporal logic approach. In *Proceedings of the 2005 International Semantic Web Conference*, 2005.
- [HTK00] D. Harel, J. Tiuryn, and D. Kozen. *Dynamic Logic*. MIT Press, 2000.
- [HV04] Z. Huang and C. Visser. Extended DIG description logic interface support for PROLOG. Deliverable D3.4.1.2, SEKT, 2004.
- [PT91] S. Passy and T. Tinchev. An essay in combinatory dynamic logic. *Information and Computation*, 93(2):263–332, 1991.
- [vB95] J. van Benthem. Temporal logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4, pages 241–350. Oxford, Clarendon Press, 1995.
- [Wie05a] J. Wielemaker. The swi-prolog rdf parser, <http://www.swi-prolog.org/packages/rdf2pl.html>. Technical report, 2005.
- [Wie05b] J. Wielemaker. Swi-prolog/xpce semantic web library, <http://www.swi-prolog.org/packages/semweb.html>. Technical report, 2005.
- [WSW03] J. Wielemaker, G. Schreiber, and B. J. Wielinga. Prolog-based infrastructure for rdf: Scalability and performance. In *Proceedings of the 2003 International Semantic Web Conference*, pages 644–658, 2003.