VERSION ANALYSIS FOR FAULT DETECTION IN OWL ONTOLOGIES

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF MASTER OF PHILOSOPHY
IN THE FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

2016

By
Maria Copeland
School of Computer Science

Contents

List of Tables

List of Figures

Abstract

VERSION ANALYSIS FOR FAULT DETECTION IN OWL ONTOLOGIES
Maria Copeland
A thesis submitted to the University of Manchester
for the degree of Master of Philosophy, 2016

Understanding changes in an ontology is becoming an active topic of interest to ontology engineers because of the increasing number of requirements to better support and maintain large collaborative ontologies. Ontology support and debugging mechanisms have mainly addressed errors in ontologies derived from reasoning tasks such as checking concept satisfiability and ontology consistency. Although debugging and tools to help the understanding of entailments have been introduced in the past decade, see [1, 2], these do not address the desirability and expectations of the entailments. Currently, logical faults in ontologies are treated in a vacuum approach that does not take into consideration the information available regarding the entailment evolution of the ontology as recorded in ontology versions, the expectation of entailments, and how the ontology and its logical consequences comply with historical changes. In this thesis we present a novel approach for detecting logical warnings that are directly linked to the desirability and expectation of entailments as recorded in the ontology 's versions. We first introduce methods for evaluating ontology evolution trends, editing dynamics, and identify versions that correspond to areas of major change in the ontology. This lifetime view of the ontology gives background information regarding the growth and change of the ontology from an axiom centric perspective and their entailment presence through out the studied versions. We then subject the asserted axioms from each version to a cross-functional and systematic analyses of changes, the effectiveness of these changes, and the consistency of these changes in future versions. From this detailed axiom change record and their entailment profiles, we derived entailment warnings that indicate or suggest domain modelling bugs in terms of content redundancy, regression, refactoring, and thrashing.

We validate and confirm these methods by analysing a ten year evolution period of

the National Cancer Research ontology NCIt. We present a detailed entailment report for each of the problematic axioms that contain domain modelling bugs, and provide a clear summary of the versions where these axioms introduce logical warnings. This detailed report of entailment history and the detection of domain modelling bugs is done without in-depth domain knowledge and purely derived from the publicly available versions of the ontology. It is through this distinctive usage of ontology versions that we pioneer the detection of domain modelling bugs as logical warnings based on the evaluation of expected and wanted entailments.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

The work and learning journey of the past four years couldn't have been possible without the constant support and encouragement of my supervisors, Dr. Bijan Parsia, and Prof. Robert Stevens. It is in their dedication and love for research that I find constant inspiration professionally and personally.

I also want to acknowledge the love and support of my family, in particular of my husband Bill. His dedication and understanding through the many challenges of conducting research has helped me reach the finish line.

I dedicate this work to my son William who shows me everyday the important things in life such as playing video games and eating pizza on movie night.

Finally, I wish to thank Prof. Uli Sattler, Dr. Chiara Del Vescovo, Dr. Samantha Bail, Dr. Nicolas Matentzoglu and Dr. Rafael Goncalvez for their contributions to this research and insight to the area of ontology evolution and support.

Chapter 1

Introduction

Ontologies are important tools for the management of domain knowledge. These logic driven knowledge-bases allow for the definition of *concepts* via the creation of *axioms* and the linkage of these concepts to explain the *relationships* between them. Depending on the ontology language and its underlying logic formalism, new knowledge from the asserted content can be inferred when a reasoner is applied ([3]). With the standardisation of OWL in 2004 as the Web Ontology Language ([4]), OWL ontologies have grown in popularity and complexity. Large collaborative OWL ontologies have emerged, specially in the domains of Medical and Bio-Informatics Sciences, such as: the Systematized Nomenclature of Medicine Clinical Terms (SNOMED CT)([5]), Gene Ontology ([6]), the National Cancer Institute Thesaurus (NCIt) ([7]), and the Foundational Model of Anatomy (FMA) ([8]).

It is evident from the number of recent publications that initial questions of 'how to best model knowledge' have become less prominent and new questions on 'how to support knowledge evolution' have risen. Tools for version management ([9]), difference analysis between versions ([10, 9]), and fault detection and explanations ([11, 1, 2]) are now a frequent part of publications, workshops, and special journal editions. The detection of errors in ontologies is still primarily handled by reasoners, thus making them the main evaluation tool available for ontologies. The evaluation and result of the reasoner tasks examine the logical correctness of the modelled knowledge and indicate problems that causes contradictions in the ontology. Reasoners highlight in their output *unsatisfiable* concepts in the ontology and render the ontology *inconsistent* when no models of the knowledge can be derived. Repair of these errors require the identification of 'broken' entailments and their removal or repair. Support for understanding these 'broken' representations has emerged in the form of justifications ([11, 1]), and antipattern detection tools ([12, 13]). Unfortunately, reasoners and these comprehension aid tools do not take into account the context in which the ontology is authored. A concept in the ontology may be properly constructed from a logic point of view (thus, no inconsistencies are present) but, it may not adhere to the expected functionality of the ontology.

Ontologies are produced in a structured life cycle that is either formalised and part of an engineering practice, or informal where the creation and deployment of the ontology happens organically. In either form, information about the ontology and its represented knowledge begins before the actual 'code' is written. During the pre-modelling phases (feasibility study, domain analysis, and conceptualisation ([14])), documentation about the domain, the ontology users and the expected functionality is created. In some ontology projects these pre-modelling documents become part of the released information for the ontology. This tends to be the case when an ontology is created under a formal engineering process. However, when an ontology is developed in an informal life cycle, these documents are usually not available externally or internally. Even in the best case scenario where documentation is released with the ontology, this information is often unreliable due to problems around the accuracy and completeness of the documentation. Similar problems with the accuracy of documentation are found in Software Engineering ([15]). Some of the reasons behind this phenomenon include the cost and time it takes to produce documentation, the knowledge of the author, and the evaluation of the final documentation ([15]). The same challenges with documentation are also found in the post-modelling phases. In these phases (implementation, evaluation, and evolution ([14])), the availability of test plans, test results, maintenance and release plans, and maturity support is incomplete or non-existent. From these observations, it is safe to conclude that documentation recording the ontology 's development and growth that can be safely relied on is the ontology 's version files.

It is from these historical version files that information about the 'expected' behaviour of the ontology can be extrapolated with some level of confidence. Ontology versions provide a record of changes, change patterns, and the consequences of these changes in the lifetime of the ontology. Exploiting this rich resource of evolutionary history not only allows anecdotal accounts of the lifetime of the ontology, but it can indicate expected and wanted behaviour from the ontology based on the consistent presence of axioms, their editing patterns, and the effectiveness of their changes. The research and results presented in this thesis test this hypothesis and provide a unique method for expanding the detection of faults in OWL ontologies. We combine techniques to statistically profile the change and growth of the ontology, often used in this type of research ([16, 10, 17]), and use them to detail interrupted entailment presence in the ontology that may suggest misalignment with the desirability of these entailments. These anomalous entailments are then subjected to cross-functional analysis of their editing patterns and the effectiveness of these changes across all versions on which these axioms appear. The sequences of these editing patterns reveal domain modelling bugs from which we produce definitions for logical warnings that indicate or suggest content regression, redundancy, factoring or thrashing.

1.1 Research Contributions

The research contributions in this thesis are detailed in the proceeding chapters. Each chapter of this thesis contributes to the body of knowledge for fault detection in OWL ontologies by providing a unique perspective, usage, and the introduction of new methods for detecting domain modelling errors based on diachronic analysis of OWL ontologies. A detailed summary of the contributions found in each chapter are the following:

Chapter 2 We explore ontologies as complex Knowledge Management Systems that require an engineering approach for their development, deployment, and maintenance. This draws significance to the documentation about the ontology 's requirements that are of equal importance as the archived versions. These documents dictate the expectations and services the ontology must provide. We focus this argument on OWL ontologies and postulate, from a fault detection perspective, that in order to support faults that are not handled by reasoners, we must integrate other available information about the ontology to detect and repair unwanted and unexpected entailments.

Chapter 3 The underlying statistical techniques, database representation of the ontology, and assumptions about 'normal' editing behaviour are described in this chapter. This chapter combines diachronic analysis techniques to produce insights about the 'hidden' information regarding entailment consistency and expectations which have not been discussed in the field of error detection.

Chapter 4 This chapter explores: (i) mechanisms to select test areas for axioms with fluctuating entailments; (ii) analysis of their change sequences and the effectiveness of these changes; (iii) the extracted editing sequences and types; (iv) definitions for domain modelling bugs; (v) and the creation of entailment profiles for problematic axioms. It is here where we explain and provide strong evidence that ontology versions can provide meaningful information regarding entailment expectations and desirability. The identification and definition of logical warnings produced in this chapter, show the importance of entailment evaluation from a functionality and expectation viewpoint.

Chapter 5 A concise evaluation of the experiments, results, key findings, and contributions to the area of ontology evaluation and evolution are discussed in this chapter. We stress the importance of this research in the area of fault detection as the first published attempt to evaluate ontologies not only from their logical construction but from their expected purpose and use. Future research and applications of this research are detailed in the Future Work Section, which demonstrates how this is only the beginning of new type of tools and services available for ontology fault detection and repair.

1.2 Published Work

The contributions of this thesis are supported by the following publications:

- [18] Maria Copeland, Andy Brown, Helen E Parkinson, Robert Stevens, and James Malone. The SWO project: A Case Study for Applying Agile Ontology Engineering Methods for Community Driven Ontologies. In International Conference on Biomedical Ontology - ICBO, 2012.
- [19] Maria Copeland, Rafael S Gonçalves, Bijan Parsia, Uli Sattler, and Robert Stevens. Finding fault: Detecting issues in a versioned ontology. In The Semantic Web: ESWC 2013 Satellite Events, pages 113-124. Springer, 2013.

Chapter 2

Preliminaries

This chapter introduces key concepts and a detailed study of Ontology Engineering, in particular the engineering process of OWL ontologies. This study provides a retrospective critique of the challenges and opportunities not previously discussed in the areas of ontology maintenance and evolution. In particular, it focuses on the opportunity to expand logical axioms support for entailment faults that are not necessarily ‘wrong’ from a formal logic perspective but may be faulty from a functional perspective. This opportunity is analysed within the context of Ontology Engineering, its lifecycle, the OWL language, and the current understanding of faults in OWL ontologies. This chapter discusses ontologies not only as formal tools for the representation of knowledge, but as a complex systems where information about its requirements, design approach, and ontological commitment are integral parts of the engineering process. These underutilised ontology resources provide important information about the expectation and desirability of entailments that can be exploited to aid fault detection and prevention as part of ontology maintenance and evolution.

2.1 Ontology Engineering

Research evidence suggests that Ontology Engineering is reaching a maturity level judged by the number of publications on ontology usage surveys, methodologies, evaluation tools, languages, and design patterns found in published literature for the past 30 years ([20, 14, 21, 22]). This field began with the need to understand, capture, represent, and interpret knowledge, which has been an area of philosophical interest since ancient Greece. It is no surprise that such a complex yet essential concept, knowledge, has become a key research area in our computerised scientific time. In Artificial Intelligence, understanding the conceptualisation of knowledge and how it can be best represented in human and computer formats has driven research since early 1980s especially in the area of natural language processing. This initial drive began with the simple question of how to acquire knowledge in a consistent way ([23]) by implementing standard processes and artefacts including ontologies. It is in this early use that definitions were presented in order to clarify ontologies ' purpose and scope. The basic definitions of ontologies that have become part of our current understanding of ontologies can be traced back to three major fields: Philosophy, Artificial Intelligence, and Knowledge Representation ([24]). It is in these three fields that the following definitions for ontologies are commonly presented:

1. *Philosophy:* "The branch of metaphysics dealing with the nature of being" ([25]). An Ontology in this sense deals with existence of knowledge rather than "how we know knowledge".

2. *Artificial Intelligence:* "An ontology is an explicit specification of a conceptualisation" ([26]).

3. *Knowledge Representation:* "A theory of vocabulary/concepts used in building artificial systems" ([27])

All three definitions deal with the theory of understanding and using a body of knowledge. This body of knowledge, when presented in a formally represented formalism, becomes a conceptualisation that captures the objects, concepts, relationships, and other entities that are assumed to exist and form an abstract, simplified view of the world ([28, 26]). A commitment to this abstract view of the world is kept when agents, humans or computers, act and use this represented knowledge in a consistent and coherent way. The means of representing this conceptualisation in explicit form is an ontology. The information or knowledge captured within the conceptualisation is known as the universe or domain of discourse. The universe of discourse captured in an ontology has definitions that associate names of entities with human and computer formats, axioms that constrain the interpretation of the universe of discourse, and well-formed use of these entities and axioms ([26]).

Ontologies are more than hierarchies and conservative definitions; an ontology is an explicit description of a domain that it is limited in interpretation depending on the defined axioms about the terms that form part of the universe of discourse. Within the fabrics of ontologies, there is engagement with allowable interpretations accessible via the use of queries and assertions that are exchanged among agents. These agents ' interactions with ontologies are conducted in an Open World Assumption (OWA) where the knowledge not represented is neither true or false, it is simply unknown. Commitment to an ontology is a guarantee of consistency of the represented knowledge but not completeness of the representation. There is also the 'creation' or realisation of new knowledge within the fabrics of ontologies. This new knowledge can be derived from an ontology through complex calculations based on mathematical logic theory performed by ontology reasoners. The use of reasoners in ontologies have allowed the confirmation of experts ' views of the world represented in a ontology, and to extract knowledge that was not explicitly stated in the ontology during the design, but emerges through the inference of knowledge through logical deduction from its asserted axioms.

The use of ontologies, not only a means of knowledge representation but as instruments to understand and discover new knowledge, has created an environment where questions about the nature of knowledge and how to elicit this knowledge have become intertwined parts of ontology design. Thomas R. Gruber in ([26]) argues that there needs to be a disciplined approach when designing ontologies. In this study Gruber presents ontologies as design decisions ([26]) and in order to properly design, evaluate, and use them there must be a shared understanding of guiding principles for the creation of the conceptualisation and the engineering of the resulting artefact ([26]). This argument for an engineering discipline is shared among other researchers at this period (see [28, 27, 24]) and provides a case for considering Ontology Engineering as a distinct field.

2.1.1 Building a case for Ontology Engineering

Early literature on Ontology Engineering has shown that ontologies are seen and used a means to establish agreements about knowledge and to provide consistency when soliciting knowledge ([28, 26, 27, 24]). The nature of ontologies requires the knowledge to be made explicit and to be able to represent this knowledge in formalisms that can capture an abstract view of the domain. This implies that there is a requirement for technology, principles, processes, and guidelines that can capture, model, reason, and communicate this knowledge. Although Ontology Engineering is not formally defined in the literature that it is now considered agreed as formal definitions for ontology (see [29, 26]), these papers do provide design criteria that aim to focus and assist with quality guidance when building ontologies. This criteria can be seen as the early attempts to establish a disciplined way of constructing ontologies which later become part of Ontology Engineering. The following list of guidelines presents the design criteria proposed in the 1990s and which are still relevant today:

1. Clarity and objectivity. Entities modelled in the ontology should provide objective definitions and natural language documentation ([26])
2. Completeness. Definitions that are necessary and sufficient are preferred over partial definitions ([26])
3. Coherence and commitment to the ontology must be kept ([26])
4. Extendibility ([26])
5. Maximise monotonic extendibility. When new terms are included in the ontology it must be done in such a way that no or limited revision takes place ([26])
6. Minimal ontological commitments to ensure agents that use the ontology have freedom to specialise and instantiate the ontology as required ([26])
7. Ontological distinction principle. Classes defined in the ontology must be disjointed ([30])
8. Diversification of hierarchies. This is to allow for multiple inheritances ([31])
9. Minimise the semantic distance between sibling concepts ([31])
10. Standardisation of names ([31])

Ontology Engineering, like Software Engineering, occurs in a life cycle where the ontology is produced, released, maintained, and finally retired. These phases can be defined as: Conceptualisation, Design, Implementation, and Deployment phases (see [32, 33]). Key activities and outcomes are specified for each of the life cycle phases. These activities give the engineers a design rationale for the ontology, help define the essential concepts of the universe of discourse, allow for a more disciplined design methodology that can be replicated and validated, and enables knowledge accumulation not only of the ontology itself but of the processes that lead to the ontology.

2.1.2 Ontology Engineering Life Cycle

In order to achieve the guidelines presented in the previous section, ontology engineers have introduced life cycle phases and activities to manage the engineering process. This section constructs a generalised picture of the recurrent life cycle phases of an ontology engineering project that can be seen as the core stages applicable to most ontologies development efforts. Much discussion has taken place in literature around the importance of building ontologies in the design phases (see [34, 35, 26, 36, 37, 24, 14]), ignoring the importance of the latter phases around evaluation, management, and evolution of ontologies. As the field of Ontology Engineering matures and greater emphasis is given to the maintenance of ontologies, important questions are being raised about the desirability and expectation of the ontology 's services, compliance with functional and technical requirements in future releases, and the integrity of version control. It is with this understanding of expected functionality that a clear baseline of expectations can be agreed for every new version of the ontology. To achieve these baselines of expectations and desirable functionalities, it is important to use diachronic analyses of project documentation, log files, and archived versions to see the functionality and desired behaviour that has persisted throughout the lifetime of the ontology.

It is with this new emphasis on an integrated approach of using project documentation and ontology evolution information (log files, version files, release notes) as part of the evaluation of ontologies that we introduce the life cycle phases and activities. The following list combines the work of [32, 14, 33] to produce the life cycle phases, activities, and an assessment on whether or not the corresponding activity documentation can be used for the evaluation of future versions of the ontology:

Table 2.1: Ontology Engineering Life Cycle Phases and Activities

Phase	Activity	Useful for Evaluation
Feasibility Study	Stakeholder analysis	No
	Purpose and scope	Yes
	Problems and opportunities	No
	Economic feasibility	No
	Potential solutions	No
	Resource allocation	No
Domain Analysis	Requirements specifications	Yes
	Motivating scenarios	No
	Competency questions	Yes
	Knowledge acquisition	No
	Solutions analysis	No
Conceptualisation	Create semi-formal ontology descriptions	Yes
	Produce architectural design of main concepts	Yes
	Design integration with existing solutions	Yes
	Evaluate the semi-formal ontology	No
	Produce a formal model of the ontology	No
Implementation	Implement the formal model in a representation language	No
	Check representation principles and reasoning exploitation	Yes
	Produce unit tests of the formal ontology	Yes
Evaluation	Reasoning focused evaluation	Yes
	Technology focused evaluation	Yes
	Stakeholder focused evaluation	Yes
	User focused evaluation	Yes
Application and Evolution	Deploy ontology	Yes (versions, log files)
	Produce evolution strategy	Yes
	Produce maintenance strategy	Yes (release notes)

The identification of phases and activities unfortunately has not guaranteed a standard approach for documenting the functional requirements of ontologies. The requirements, scope, and intended functionality produced in the Feasibility Study, Domain Analysis, and Conceptualisation phases are usually poorly documented and does not reflect the state and purpose of the ontology in later releases. This lack of continuity of documented requirements and expectations results in informal ad-hoc workarounds and hacks to the ontology in order to support the ontology in later versions ([14]). It is in this light, that the latest releases of the ontology can be seen as a proxy of the requirements and expectations that the ontology must meet. Taking into account the reality that functional requirements and official documentation cannot stay up to date as the ontology evolves, it is pragmatic to suggest that the 'current' state of expectations and functionality of the ontology is best captured in its latest released version. This conclusion and assessment of the current state of documentation accuracy not only point to the need for better documentation management, but also for better version archiving and management tools to allow for retrospective studies of requirements, desired functionality, and expected services.

This section has stated the importance of a guided and disciplined approach for the development, release, and management of ontologies in general. It has drawn attention to the often ignored documentation in an ontology such as versions, log files, and release notes which are very important in the evaluation of expectations and functionality of the current version of the ontology. In the next sections, the focus shifts to OWL ontologies, the logic underpinnings, profiles, and common errors found in OWL ontologies that serve as the foundations for the remaining of this thesis.

2.2 OWL Ontologies

2.2.1 Ontology Languages and Deduction Services

An ontology language is a formally constructed set of language symbols constrained by specific logic rules that are used to build elements and statements that express a domain 's representation ([38]). The statements written by the elements in the ontology language define *concepts, roles, instances, and axioms*. The union of these sets of elements is refered to as the knowledge base or *ontology*, denoted as O. These sets of elements expressed in the ontology are commonly refered to as the *signature* of the ontology represented by the symbol $\widetilde{O}$ ([39]). Ontology languages map to a logic formalism, commonly based on Description Logic DL or First-order Logic FOL, which supports formal semantics and reasoning services with the help of deduction systems ([38]). Meaning, or *interpretation* I, is assigned to the sets in O by defining a non-empty domain Δ^I and a function $\cdot^I$. The interpretation I is a *model* of O if every axiom is consistent in O. When this is the case, the interpretation entails the ontology; in other words, the ontology is a logical consequence of the interpretation, denoted as $I \models O$.

A major strength of ontologies as Knowledge Systems is the ability to provide deduction services when applying decision procedures via reasoners. These services extract implicit knowledge from the ontology, inferred axioms, which are not explicitly stated by the ontology developers, but are useful to developers as means to expand and discover knowledge.

The types of deductions problems reasoners handle when applied to ontologies are:

- *Membership (Realisation):* This applies to atomic concept membership and it infers whether an individual is an instance of the concept. This can only be achieved if the individual is an instance of all the models of the concept ([38]). That is, given an atomic concept C in an ontology O and it is an element of $\widetilde{O}$, an instance a is an instantiation of C if and only if $O \models C(a)$.

- *Classification:* In classification the reasoner infers all the subconcept and superconcept relationships between asserted and inferred atomic concepts ([40]). This is achieved by testing entailments of the form $O \models A \sqsubseteq B$, where A, B are atomic concepts. The resulting classification hierarchy is depicted as the classification tree.

- *Equivalence:* It deals with all equal extensions between atomic concepts, $O \models A \equiv B$, where A, B are atomic concepts ([38]). This reasoner service reduces any redundancy in O since it discovers equivalent descriptions and reuses concepts.

- *Concept Satisfiability:* This checks whether there are any empty extensions of a concept ([38]). That is, a concept A is unsatisfied in O if there are no models of A in O, denoted as $O \models A \sqsubseteq \bot$. This can result when there is a contradiction in the use of a concept in the ontology.

- *Ontology Consistency:* An ontology O is consistent if there is at least one valid model for the interpretation function I, $I \models O$ ([38]).

- *Entailment:* Entailment checks whether an axiom α is a logic consequence of O, denoted as $O \models \alpha$ ([40]).

In addition to these services, some reasoners provide the following non-standard inference services to aid in the building and presentation of the ontology:

- *Concept approximation.* This inference service has mainly been researched for achieving translation and inference problems in expressive and inexpressive DLs ([40]). This service is useful for approximating reasoning and for presenting comprehensive presentations of the ontologies to non-expert uses ([38]).

- *Justifications or explanations for entailments.* A justification $\mathcal{J}$ for an entailed axiom α corresponds to the minimal subset of axioms in an ontology O for α to hold. This is denoted as $\mathcal{J} \subseteq O$, $\mathcal{J} \models \alpha$ [11].

- *Ontology modularity.* This service assists in identifying parts of an ontology O, called modules, that capture all that is said in O over a given set of terms Σ, called a signature ([41]). This allows users and developers to work with smaller sets of the ontology instead of the dealing with the entire ontology. Commonly used procedures to identifying modules are based on semantic or syntactic locality [42]. The Atomic Decomposition (AD) as defined in [43], allows the representation and visualisation of the modular structure of O by extracting the basic components, called atoms, of modules and their relationship. This service is particularly helpful in ontology comprehension and modular reasoning.

The use of the standard and non-standard reasoning services, the choice between computational versus expressiveness of an ontology, and requirements on formal semantics bring forward the need for a careful selection of DL based ontology languages when designing ontologies. In the next section, we discuss Ontology Web Language OWL and the different profiles of OWL.

2.2.2 OWL - Web Ontology Language

In early 2000 's The World Wide Web Consortium identified the need for a more expressive language for web ontologies than RDF. This language requirement built on the work carried out by research groups in America and Europe, which had previously identified a powerful ontology modelling language called *DAML+OIL* ([44]). In February 2004, the Ontology Web Language (OWL) became the standardised recommendation for ontology language for the Semantic Web [4]. This version of OWL extends $\mathcal{ALC}$ semantics (atomic negation, concept intersection, universal restrictions, existential quantification, and complex concept negation) to included transitive role hierarchies ($\mathcal{SH}$), nominals (O), inverse roles (I), and cardinality restrictions (N); resulting in $\mathcal{SHOIN}$ DL.

In 2004, the W3C 's Web Ontology Working Group defined OWL with three sub-languages that can properly accommodate the different requirements of expressiveness and automated reasoning: OWL Full, OWL DL, and OWL Lite (see [45, 46]). In a survey of OWL ontologies in the Web by Matentzoglu in ([47]) the OWL DL profile is the most widely used as judged by the number of OWL DL ontologies in the publicly available ontology repositories such as BioPortal and TONES Repository. Standard reasoning services have been successfully implemented for OWL DL in several reasoners such as FaCT++, Pellet, RACER, and KAON2 ([48]).

In 2009, The World Wide Web Consortium introduced a second iteration of OWL known as OWL 2. OWL 2 ontologies can be stored as Semantic Web documents with a primary exchange between OWL2 and RDF/XML syntax ([49]). It uses Direct Semantics and RDF-Based Semantics, which assign meaning directly to the ontology structures and RDF graphs ([49]). OWL 2 also provides sub-languages, called profiles, which allow flexibility depending on the modelling and application scenarios ([49]). These profiles are:

- OWL 2 DL: it is based on $\mathcal{SROIQ}$ DL allowing qualified cardinality restriction ([45]).
- OWL 2 EL: it is based on $\mathcal{EL}$ DL which allows concept intersection and existential restrictions. This enables the execution of reasoning tasks in polynomial time; suited for large ontologies where expressive power is important to the ontology implementation ([45]).
- OWL 2 QL: it is part of the OWL-Lite DL family. It allows for conjunctive queries using standard relational database technology. This architecture model allows for ontologies that are very light in expressiveness, store large numbers of instances, and the ontology purpose is to primarily answer relational/database type queries ([45]).
- OWL 2 RL: it is an extension of OWL 2 DL with RDFS. It has polynomial time reasoning and uses rule-extended database models which operates directly on RDF triples. Like OWL 2 QL, this language is suitable for lightweight ontologies with large numbers of instances with the additional benefit of operating directly on RDF triples data ([45]).

In addition to logical components (axioms, classes, instances, and objects) that define the domain in OWL ontologies, ontologies can be associated with additional information or documentation via the use of annotating axioms ([45]). These non-logical axioms provide a means to capture information, or meta-data, about the ontology, classes, axioms, and instances. Although these axioms do not add logical meaning to the ontology, they serve an important role by providing human readable information about the domain, the ontology version, and the evolution process.

2.3 Faults in OWL Ontologies

With the expressiveness advantages with OWL 2 and the increasing use of OWL ontologies, it is no surprise that much of OWL ontologies research focuses on preventing and repairing faults in ontologies ([50, 13, 51, 52, 2]). This thesis defines faults in ontologies as:

Definition 2.1. A **fault** is a deviation from required and expected behaviour in ontology O, where behaviour is akin to service and the effectiveness in the delivery of this service ([19]).

In Software Engineering, faults can be divided as functional (service) and non-functional (delivery of service) faults. The severity of these faults depend on the impact to the system and the service it is expected to provide. A complete system crash, that is ceasing to work completely, is considered a high severity fault and its impact is major due to the complete disruption of service. Equally, a fault that deviates entirely from the expected behaviour and compromises the validity of the outcome is categorised as a high severity fault even though the system is available. Arguably, a system crash is an immediate indication of a severe fault in a system; where as a non-functional fault may be harder to identify and in some case diagnose. Thus, proper identification, diagnosis and system repairs rely entirely on technical and domain knowledge of the service in order to tackle the complex area of faults management.

Similar requirements and challenges in fault identification and repair are found in Ontology Engineering. In ontologies, faults are related to lack of modelling consistency, insufficient adherence to the ontological commitment of the domain, errors in the asserted knowledge defined in the ontology, and the wrong or non-logical consequences of the asserted knowledge. Faults may be introduced either by human intervention (e.g. modelling decisions, lack of domain knowledge, etc.) or tools intervention (e.g. automatic content generation tools, ontology merging tools, etc.). Ontology faults ’ relation to logical formalism, design and implementation strategies make for a natural classification of *logical* and *non-logical* faults. *Logical faults* deal directly with the logic driven aspects of the ontology; specifically the asserted and entailed knowledge of ontology. In Software Engineering terminology, these faults are primarily functional faults as it deals directly with the logical service the ontology is expected to provide. On the other hand, *non-logical faults* deal with the 'how' the knowledge is modelled and documented in the ontology. Modelling decisions regarding patterns, specificity of concept definitions, information about the ontology in annotation axioms, and ontological commitment are directly related to non-logical, or non-functional, aspects of the ontology.

Techniques and tools for detection, diagnosis, and repair of OWL ontologies depend on whether faults are either logical or non-logical in nature. Reasoners such as $FaCT++$ and $Pellet$ ([53, 54]) detect logical errors by applying reasoning tasks that deal directly with entailments and the consistency check of the ontology. In addition to reasoners, debugging logical faults can be achieved with justifications where a subset of the ontology is presented showing the explanations why an entailment holds ([10, 50, 13, 11]), ontology completion where inferred axioms are suggested as additions to the explicit knowledge of the ontology ([3]), and difference analysis between two OWL files of the same ontology ([55, 56]). Non-logic faults can be detected by irregularities in modelling patterns ([57]), performance problems ([58]), and any errors dealing with the information recorded in the annotation axioms. Logical faults are primarily dealt by automatic tools that are add-ons to IDEs that identify and allow for repairing the logical fault in the ontology authoring environment. In contrast, most non-logical faults may require a combination of automatic detection tools and human driven quality assessment process to detect and repair these errors in the ontology.

The remainder of this chapter focuses on the logical aspect of the ontology, that is, the identification and debugging support of entailment faults, asserted and inferred, for OWL ontologies. We look directly at problems with entailments and their justifications, differences that indicate errors throughout the evolution of the ontology, common entailment faults detectable by reasoners, and entailment faults that require analysis, understanding, and tools outside the established ontology reasoners.

2.3.1 Entailment Faults in OWL Ontologies

The scope of this thesis is restricted to the *logical* behaviour of the ontology; that is, the set of entailments retrievable by standard reasoners. Debugging tools that apply reasoners detect entailment errors when the outcome of reasoner tasks indicate contradictions, incoherence and inconsistency in the declaration of classes and the construction of axioms.

2.3.1.1 Unsatisfiable Classes

Unsatisfiable classes indicate a contradiction in the definition and use of a class. This error is most likely a modelling error caused either by misinterpretation of the domain from the domain expert, or an implementation error from the ontology developer. When unsatisfiable classes are instantiated, the ontology becomes inconsistent, thus indicating a high severity fault due to the 'uselessness' of the logical inferences of the ontology. If an unsatisfiable class is not instantiated, the ontology can still have valid models making the ontology incoherent. An incoherent ontology maybe harder to debug if the unsatisfiable classes are not made visible by the debugging tools. Fortunately, debugging tools usually indicate unsatisfiable classes either by highlighting them in red or showing them as subclasses of *bottom*.

2.3.1.2 Tautological Classes

A tautological class is a class entailed as a superclass of all the classes in the ontology, and where all individuals are instances of the class. In ([1]) the authors explained that this type of fault is usually an unintended and unnoticed consequence without the ontology users understanding why it is present in the ontology. Debugging tools may or may not indicate tautological classes as errors thus making its identification and diagnosis harder for the ontology developer ([1]).

2.3.1.3 Ontology Inconsistency

An ontology is inconsistent when no models are inferred by the reasoner. The ontology provides no meaningful knowledge thus disrupting completely the service it is meant to provide. Inconsistent ontologies are high severity faults akin to a complete system crash in Software Engineering.

The identification of these errors is achieved when the reasoner cannot confirm that the entailments hold in the ontology, thus triggering a signal to the debugging tools of these entailments as faults. The identification, diagnosis, and debugging of these errors based on reasoners ' services depend on the underlying formal semantics, interpretation function of the underlying logics (OWL profiles), and the ability of reasoners to indicate faulty entailments.

2.3.2 Domain Modelling Bugs

The next set of logical faults are not necessarily wrong from a formal logic perspective, but from the understanding and representation of the domain knowledge and the ontology service it is expected to provide. Specifically, these faults are dependent on the *desirability* and *intention* of the logical consequences. In order to distinguish them from the logical faults found by debugging tools that apply reasoners, this thesis introduces the notion of *domain modelling bug* ([19]).

Definition 2.2. Let $\mathcal{O}$ be an ontology and α be an axiom in $\mathcal{O}$, α is a **domain modelling bug in** $\mathcal{O}$ if $\mathcal{O} \models \alpha$ and α is not a desired entailment, or $\mathcal{O} \not\models \alpha$ and α is a desired entailment.

The *desirability of entailments* in an ontology $\mathcal{O}$ is determined by the analysis of the written expectations and intentions of services that the ontology must provide as defined by the ontology 's purpose and use. For instance, consider an OWL DL ontology $\mathcal{O}$ with a TBox $\mathcal{T}$ which contains the following asserted axioms with the resulting entailment (Example 2.3). These asserted axioms are expected to model the specifications (definitions) for University Student, Post Graduate Student, and Research Student found in Table 2.2:

Example 2.3.
$$\begin{array}{ll} \mathcal{T} = \{ & \text{PostGraduateStudent} \sqsubseteq \text{Student} \\ & \text{ResearchStudent} \sqsubseteq \text{PostGraduateStudent} \; \} \\ \text{and} & \\ \mathcal{O} \models \{ & \text{ResearchStudent} \sqsubseteq \text{Student} \; \} \end{array}$$

Table 2.2: Student Types Requirements

Student Type	Description
University Student	*University Student* is a specific type of student who is studying at a university
Post Graduate Student	*Post Graduate Student* is a kind of *University Student*
Research Student	*Research Student* is a kind of *Post Graduate Student* that is a *University Student*

The entailment $ResearchStudent \sqsubseteq Student$ is a *domain modelling bug* in this example because it is not a desired modelling consequence of the ontology. Although, logically the entailment follows from the ontology, the asserted axioms do not model the expected definitions for student types as expressed in Table 2.2. Since the asserted axiom that models Post Graduate Student is not defined in terms of University Student, and the resulting entailment defines Research Student as a general student, it is clear to see that the modelled domain and the logical consequences do not meet the expectations set in the requirements. Similarly, the missing expected entailment $ResearchStudent \sqsubseteq UniversityStudent$ is a *domain modelling bug* because it is absent from the TBox and it is not a logical consequence of the modelled knowledge.

The detection of domain logical bugs relies heavily on human expertise of the domain and modelling conventions defined at the beginning of the engineering process, and on heuristics that identify possible *undesired* consequences from the ontology. The following entailment faults can be categorised as domain modelling bugs:

2.3.2.1 Antipatterns

This type of errors occur when OWL constructors are misused or confused by the developers. In the studies ([12, 59, 2]), the authors concluded that the most common antipatterns are: (i) AND for OR, (ii) subsumptions and equivalences, (iii) not obeying defined data and property restrictions, and (iv) axioms that add no modelling support or meaningful knowledge (*dead axioms*).

Consider the following extension to Example 2.3, where an equivalence is wrongly added to model the intended statement Master Student is a subtype of (or it is subsumed by) Post Graduate Student (see Example 2.4). The misuse and misunderstanding of subsumption and equivalence make this domain modelling bug a common error that results not only in a wrong assertion ($MasterStudent \equiv PostGraduateStudent$), but it can also produce unwanted consequences such as Master Student is a subtype of

Student (*MasterStudent* $\sqsubseteq$ *Student*). As pointed out in ([12]), this error is sometimes the result of poor training and understanding of the meaning of OWL constructors.

Example 2.4.

$$\begin{array}{rl} & \text{ResearchStudent} \sqsubseteq \text{PostGraduateStudent} \\ \mathcal{T} = \{ & \text{PostGraduateStudent} \sqsubseteq \text{Student} \\ & \text{MasterStudent} \equiv \text{PostGraduateStudent}\} \\ \text{and} & \\ & \text{ResearchStudent} \sqsubseteq \text{Student} \\ \mathcal{O} \models \{ & \text{MasterStudent} \sqsubseteq \text{Student} \\ & \text{ResearchStudent} \sqsubseteq \text{MasterStudent}\} \end{array}$$

The detection of antipatterns is not a standard debugging service of OWL ontology authoring tools. An exception is the ontology editor SWOOP, which uses a heuristic-based repair tool that identifies antipatterns, and suggests their removal or modification from the ontology ([2]).

2.3.2.2 Wrong Entailments

Wrong entailments are statements in the ontology that are ‘not valid’ with respect to the ontological commitment and modelling approach of the ontology. Like antipatterns, wrong entailments are human errors derived from lack of understanding of the domain or how ontologies work. These domain modelling bugs can be the result from unresolved ambiguities in the ontology left unattended during the developing process. Due to the ontologies ’ Open World Assumption (OWA) developers and modellers must assert not only the *what is* concepts of the domain, but the *what is not* concepts too.

As seen in Example 2.4, the addition of a misused equivalence results in a new inferred axiom *ResearchStudent* $\sqsubseteq$ *MasterStudent*. If the intended service and representation is required to make the definition of Research Student and Master Student different (*disjoint*) from one another, the omission of an axiom that declares this disjointness results in the wrong entailment, or knowledge, of Research Students being a kind of Master Student.

The identification of wrong entailments can be carried out if underlying access, either manual or automatic, to the intention of the ontology is available and if it can be extrapolated what is desired and expected to declare or omit in the ontology.

2.3.2.3 Unwanted Entailments

Unwanted entailments are logical consequences that are not desired yet are present in the ontology. It could be argued that all entailment faults in an ontology can be categorised as unwanted entailments. To disambiguate from this general use, this thesis distinguishes unwanted entailments strictly as *domain logical bugs* because these deviate from the desired expectations of service of the logical consequences. These entailments deal directly with the wrong consequences of erroneously represented content and not from a faulty modelled domain such as when antipatterns are present. An example of an unwanted entailment was found in Example 2.3, where $ResearchStudent \sqsubseteq Student$ is an unwanted consequence of the wrongly modelled knowledge.

2.3.2.4 Missing Entailments

Missing entailments are expected entailments that are not elements or logical consequences of the ontology. These entailments occur when the content of the ontology does not explicitly declare the intended entailment, nor it is inferred from the knowledge modelled in the ontology. Ontology developers use the ontology to author statements about the domain and also to elicit new knowledge that can be derived from these statements. When there is an error in the representation, either because of a logical or modelling fault, an expected entailment may not be a consequence of the ontology. If we consider again the case of entailment faults due to the wrong understanding of Open World Assumption (OWA), we can easily see that axioms that do not restrict the definition of what a concept *is not* are missing entailments, and can derive wrong entailments from this missing information.

The detection of these entailment faults and domain modelling bugs require a formal and practical understanding of ontology building. When and how these faults should be debugged and repaired depends largely on the impact (high impact faults may require immediate repair, where as minor faults can be repaired at a later time), the validity of the service outcome, and the user expectations of service. As seen in this chapter, it is no longer enough to just rely on reasoners for the identification of entailment faults. The complex array of logical and modelling faults require tools outside reasoning tableaux to address entailment errors. It is equally important to provide debugging services that can access, navigate, and analyse ontology evolution, expected functionality, and intended service to tackle entailment faults that result from domain modelling bugs. This new insight puts ontologies as more than simplistic knowledge representation artefacts, where error detection approaches rely only on the ‘technical’ aspects of ontologies. Ontologies need to be seen as knowledge systems that require proper engineering approaches and a more in-depth understanding of the services ontologies provide.

2.4 Discussion

The analysis presented in this chapter shows that ontologies are complex Artificial Intelligence systems that require systematic approaches for design, management, and evolution support. The field of Ontology Engineering aims to guide and discipline the processes and tools used in the design, building, deployment, and management of ontologies. This research field is moving away from the initial focus of how to build ontologies, to focus on how to better support ontologies, specially as these become more mainstream Knowledge Management tools. The logical foundations found in DL ontologies, and the standardisation of OWL and its profiles, have provided a starting point from which reasoning services have been developed to support fault detection, debugging, and correction services. As seen in this chapter, there has been an emphasis on fault detection in OWL ontologies based on the output of reasoners. Although this is a vital service that guarantees a correct logical representation, this service is limited and does not take into account the content and modelled correctness of the representation. This single-minded approach for repairing ontologies does not consider the whole ecosystem in which ontologies are produced. If we can integrate information about the modelling decisions and historical entailment commitment, entailment fluctuations and editing patterns, and the effectiveness of changes between versions, we should be able to provide a much more comprehensive approach for repairing ontologies.

The subsequent chapters in this thesis take this conclusion as its main drive to design a novel method for fault detection of domain modelling bugs. As seen in this chapter, access and understanding of the intentions and expected service of the ontology is imperative in the detection of domain modelling bugs. The remainder of this thesis proposes a method for determining the desirability of entailments which form the basis for the identification of domain modelling bugs. This method applies diachronic studies of editing patterns, entailment presence, and analysis of the effectiveness of changes between versions to support conjectures regarding the desirability and expectation of entailments. From this analytical based approach, we are able to identify domain modelling bugs and trigger warnings for these faulty entailments. We also provide supporting information about these warnings to indicate when in the ontology 's history these entailments deviated from the expected service and use of the ontology.

Chapter 3

Methods and Materials

To achieve the expansion of fault detection services for domain modelling bugs in OWL ontologies, it is important to access and analyse the intentions and desired behaviour of the ontology. As seen in the previous discussions, the reliability and availability of documented requirements and the ontology 's engineering process is difficult to acquire due to lack of management, quality, and reliable information. In this chapter, we propose analytical methods for identifying the desirable entailments in an ontology based on the historical record of entailment presence, editing actions, and the effectiveness of such actions. We detail the experiment and infrastructure specifications for conducting this analysis, define key statistical approaches used in the study of the ontology 's lifetime record, and identify assumptions about what it is considered expected development behaviour in a growing ontology. The methods, materials, and experiment results used in this and the following chapters expand from the work published in ([19, 18]) and presented in the International Conference of Bio-Ontologies (ICBO 2012), European Semantic Web Conference (ESWC 2013), and the International Semantic Web Conference (ISWC 2014) [1] [2] [3].

3.1 Ontology Corpus Selection

A detailed study of entailment dynamics and the impact of these dynamics in an ontology requires a consistently edited and used ontology with an active user community.

[1]http://kr-med.org/icbofois2012/
[2]http://2013.eswc-conferences.org
[3]http://iswc2014.semanticweb.org

An ontology that is constantly adapting to the needs of a user community, best reflects the community 's current expectations and needs. It is through this source of recorded activity that observations about the ontology evolution and the service it provides can be deduced. A 'live' ontology should satisfy and comply with user requirements, which are usually detailed at the inception of the ontology and have evolved to meet current needs. This latest version is the most up to date source of information regarding the expectations of service the ontology must provide; whereas the previous versions contain information regarding the reliability and longevity of this service and expectations.

Ontologies that meet this criteria are usually found in large organisations with highly collaborative environments such as the U.S. National Cancer Institute and its ontology the National Cancer Institute (t)Thesaurus (NCIt) ([7]) and the Gene Ontology Consortium with its ontology the Gene Ontology (GO) ([6]). In this thesis we focus on the NCIt corpus because of its rich historical documented record of authoring and maintenance analyses (see ([60, 61, 62])), its rigorous release cycle (see ([63, 64, 65])), authoring of logical and non-logical axioms ([63, 7, 64]), use of inference services (reasoners) to entail new knowledge from the asserted axioms ([63, 64]), and monthly publicly available OWL files.

3.2 The NCIt Case Study

The NCIt ontology supports the authoring and maintenance of cancer research terminology part of the NCI research. The ontology is used a means to standardise vocabulary across the NCI and provide vocabulary resource services to both public and private organisations, such as the the Food and Drug Administration (FDA) ([66]). Documented information about its features and services are publicly available and have been evaluated internally and externally ([61]). The NCI identifies the following key features of the NCIt service: ([66]):

- Unique codes for defined concepts
- Identification of terms to include: synonyms, research codes, external codes, and known names
- Link to metadata resources
- Formal logic-based definitions

- Integration with other data sources, including the NCIt subsets

Through out its history, the NCIt has been a source of content and ontology engineering studies in the OWL community. Most notably, studies around the activity and function of the ontology have focused primarily on particular versions or most current version (see ([63, 7, 64, 62]) providing limited information regarding the general lifetime of the ontology. Only recently, studies regarding the overall growth and decline cycles of the NCIt have emerged. In the work by Gonçalves et al. ([67, 16]), the authors studied the overall evolution of the NCIt by analysing 86 versions of the ontology, corresponding to the period from 2003 to 2011, using pair-wise consecutive versions to calculate the net gain or reduction of asserted and inferred axioms, and the logical impact of the additions and removal of axioms. Although this study is a first step in identifying fluctuations in the entailments of the NCIt, its view is restricted to two consecutive versions analysis without providing insight to the lifetime presence of the entailments in the ontology. In order to provide this lifetime view of the entailments in the NCIt, it is necessary to broaden the analysis from a pair-wise approach to an all-versions approach to extract the lifetime profile of the entailments in the NCIt. This broader view provides a diachronic perspective of the entailments, which identifies entrance, presence, removal, or modification through out all the studied versions.

To achieve this broader view, the NCIt case study in this thesis includes: 112 versions of OWL files (from release 02.00 through to 13.05d), 88 log files (from monthly history 06.01c (January 2006) to 13.05d (May 2013)), and monthly released reports covering nearly a ten year period from September 2003 to May 2013. All records were obtained from the NCIt 's public website[4] and a mapping of the NCIt version names, month and year, and the version numbers used in this thesis can be found in Appendix A.

The evaluation of these 112 versions of the NCIt will produce a diachronic view of the asserted axioms lifetime in the ontology, entailment profiles for the asserted axioms that have been retired (removed) from the ontology, and the modified axioms that remain in the ontology. We can speculate the desirability and expectations of the entailments from these observations, and identify unwanted or missing entailments that do not meet the expectations of the behaviour observed in the diachronic analysis. It is through this insight and novel use of ontology evolution analysis that this thesis tests the hypothesis raised in the previous sections; namely, the detection of domain modelling bugs can be achieved by extrapolating desired expected entailments based

[4]`ftp://ftp1.nci.nih.gov/pub/cacore/EVS/NCI_Thesaurus/archive/`

on the historical presence of entailments.

3.3 Experimental Plan

The design and execution of the experiments ran against the NCIt case study cover the following areas:

1. The NCIt 's asserted axioms lifetime record from 2003 to 2013.
2. Identification of the expected editing events corresponding to the asserted axioms in a versioned ontology.
3. Identification of the NCIt 's asserted axioms that conform with the expected editing events.
4. Identification of the NCIt 's asserted axioms that deviate from the expected editing events.
5. Produce an entailment profile, which includes the logical impact to the NCIt ontology, for the axioms that have pathological editing events.
6. From these entailment profiles, define and categorise the types of domain logical bugs found.
7. Verify the findings with the NCIt authors and developers.

From this comprehensive analysis of the NCIt, we will produce general conjectures regarding fault detection methods for domain modelling bugs that rely primarily on the diachronic axiomatic analysis of the ontology.

3.3.1 Experiment Setup

The experimental setup for diachronic analysis of the NCIt s versions requires the use of systems and software for the extraction and organisation of data representing the axioms, inferences, classes, together with the corresponding mappings to the ontology files (version files). The data extraction required the creation of custom-built JAVA program that could parse the NCIt OWL version files using the OWL API v3.x ([68]). The data extracted from the ontology files corresponded to the asserted axioms and classes from the ontologies ' TBoxes, and saved as CSV files. These CSV files also contained the classification of these extracted axioms, which was achieved by the JAVA program 's execution of classification of the extracted axioms using FaCT++ v1.6.2 ([53]). No data was extracted for instances found in the ontology 's ABoxes since the analysis of the NCIt corpus focuses on axioms and their entailments. In addition to the extracted data for axioms, inferences and classes, the CVS files also contained data from the effectuality analysis we conducted for all versions in the NCIt corpus. Effectuality analysis correspond to the method developed by Gonçalves et al in ([16]) to identify and categorised the logical impact and effectiveness of axioms additions and removals between two versions of an ontology. This difference analysis (a.k.a diff analysis) uses the following notions for effectual and ineffectual additions and removals ([16]):

Definition 3.1. Let O_i and O_j be two versions of an ontology O where $i < j$. An axiom α is an *addition* if $\alpha \in O_j \setminus O_i$, and a *removal* if $\alpha \in O_i \setminus O_j$. The logical consequences between O_i and O_j are:

- Let α be an **effectual addition** if $O_i \not\models \alpha$ (denoted $\alpha \in \text{EffAdd}(O_i, O_j)$), and **ineffectual addition** otherwise (denoted $\alpha \in \text{IneffAdd}(O_i, O_j)$).
- Let α be an **effectual removal** if $O_j \not\models \alpha$ (denoted $\alpha \in \text{EffRem}(O_i, O_j)$) and **ineffectual removal** otherwise (denoted $\alpha \in \text{IneffRem}(O_i, O_j)$).

The resulting effectual and ineffectual edited axioms, together with the extracted and classified data stored in the CVS were then migrated to a MySQL v5.1.63 database via a automatic import. The data in the database was organised with the underlying schema detailed in Figure 3.1. This schema represents the following mapping between the NCIt ontology files were each NCIt version is identified as O_i and i indicates the version number.

1. Ontology O_i: Each ontology 's file name is stored in the table *Ontology* with a integer identifier i.
2. Axioms $\alpha_j \in O_i$: Each structurally distinct axiom α_j is stored in the *Axioms* table identified by j. Each axiom tuple (j, i) is stored in the table *Is In* to match each axiom by their j id as asserted in ontology i.
3. Classes $C_j \in O_i$: Each class name C_j is stored in the table *Classes* identified by j. Each class tuple (j, i) into the table *Class In*.

4. Usage of class C_j in O_i: Each class C_j used in axiom $\alpha_k \in O_i$ is stored in table *Used In* as a triple (j,k,i).

5. Atomic subsumptions (equivalences) β_j such that $O_i \models \beta$ and $\beta_j \notin O_i$: Each inferred atomic subsumption (equivalence) β_j are stored in the table *Inferred Subsumptions* (*Inferred Equivalences*). Their corresponding tuple $(j,\, i)$ is stored in the table *Inferred Subsumptions In* (*Inferred Equivalences In*).

6. Effectual changes: Each added (removed) axiom $\alpha_j \in \mathrm{EffAdd}(O_i, O_{i+1})$ ($\alpha_j \in \mathrm{EffRem}(O_i, O_{i+1})$), identified by j, is stored in the table *Effectual Additions* (*Effectual Removals*) as a tuple $(j,\, i+1)$ ([10]).

7. Ineffectual changes: Each added (removed) axiom $\alpha_j \in \mathrm{IneffAdd}(O_i, O_{i+1})$ ($\alpha_j \in \mathrm{IneffRem}(O_i, O_{i+1})$), identified by j, is stored in table *Ineffectual Additions* (*Ineffectual Removals*) as a tuple $(j,\, i)$ ([10]).

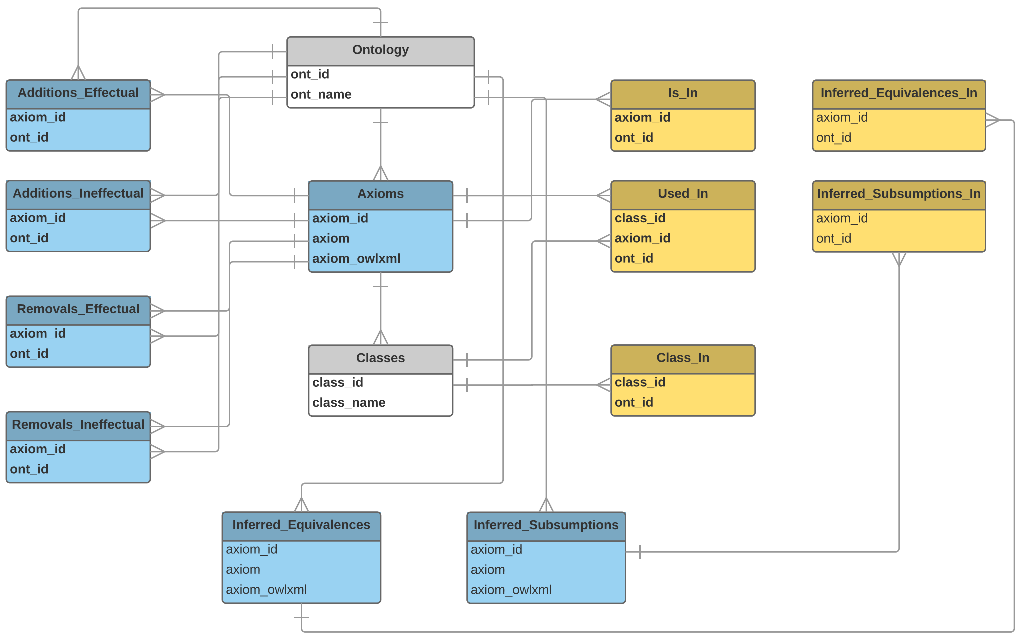


Figure 3.1: Experiment Database Schema

This experiment data setup allows for data analysis via the execution of SQL queries that apply statistical methods used for diachronic studies. A detailed description of the methods used for the SQL-query based data analysis, and subsequent difference analysis between versions of the NCIt ontology is described in the next section.

3.3.2 Experiment Methods

The experiment methods applied to the set detailed in the previous section include statistical analysis of the axiomatic lifetime evolution, axiomatic presence, and this presence 's distribution across all versions in the NCIt corpus. From here, we will be able to analyse any possible correlations between the results and our definitions of expected editing patterns found in a large collaborative ontologies. Any observations that deviate from this expected editing behaviour, will be analysed in further detail to produce an entailment profile that indicates all additions and removals, and the logical impact of these actions to the NCIt.

To begin the statistical lifetime analysis of the NCIt, we used a Time Series analysis to provide a full view of the NCIt lifetime evolution from 2003 to 2013. Our Time Series data corresponds to the successive count of the asserted axioms for each version of the NCIt for the ten year period. The Time Series analysis will be used to identify growth or declines periods and highlight any patterns to these observations. Once this overview of the evolution is produced, a Frequency Distribution analysis will be applied. In this statistical analysis, the frequency, or count, is generated for the number of versions an axiom is present in the corpus. The resulting frequency distribution table will show each axiom count as a mutually exclusive data point. To graphically explain these results, we will use a histogram diagram to graph each data point (axiom count) and sort them from highest to lowest frequency. This will allow us to identify the stability of axioms judge by their presence in consecutive versions of the corpus.

In order to judge whether a deviation from the expected service in an ontology has taken place, it is important to define the *expected* editing actions and logical consequences of these actions in an active ontology. In an active ontology, it is natural to assume that axioms are added, edited, and/or retired over the lifetime of the ontology. These changes occur as the knowledge evolves and modifications, additions, and removals take place on what was previously asserted in the ontology. Each version of the ontology is a snap shot of the knowledge captured at that point in time, and any changes to that knowledge is captured in future versions. From this understanding of knowledge evolution in a versioned ontology, it can be assumed that the following editing behaviours and logical consequences are standard in an active ontology ([19]):

Definition 3.2. Let O_i be a version of an ontology O and the axiom α be an addition in O_i ($\alpha \in O_i \setminus O_{i-1}$), α has **consecutive entailment presence** in O if $O_{i..n} \models \alpha$, where $i < n$ and i is the first version α is entailed, and n is the last version where α is entailed.

Definition 3.3. Let O_j be a version of an ontology O and the axiom α be a removal in O_j ($\alpha \in O_{j-1} \setminus O_j$), α has **removed entailment presence** in O if $O_{i..j-1} \models \alpha$ and $O_{j..z} \not\models \alpha$, where $i < j < z$ and i is the first version α is entailed, j is the last version where α is entailed, and z is the last version of O.

Definition 3.4. Let O_k be a version of an ontology O and the axiom α be a modification in O_k, where a *modification* is a change to α such that $\alpha \notin O_k$ and $O_k \models \alpha$. The axiom α has **consecutive entailment presence following a modification** in O if $O_{i..n} \models \alpha$, where $i < n$ and i is the first version α is entailed and n is the last version where α is entailed.

The Figures 3.2, 3.3, and 3.4 can best illustrate the definitions presented above. The NCIt versions are visualised as points in a line that represents the lifetime of the ontology, and the entailment presence for an axiom α in each version is represented as a full circle for an assertion and a dotted circle for an inference.

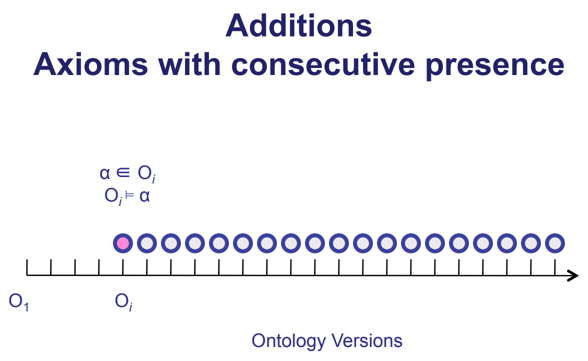


Figure 3.2: Addition - Consecutive Entailment Presence

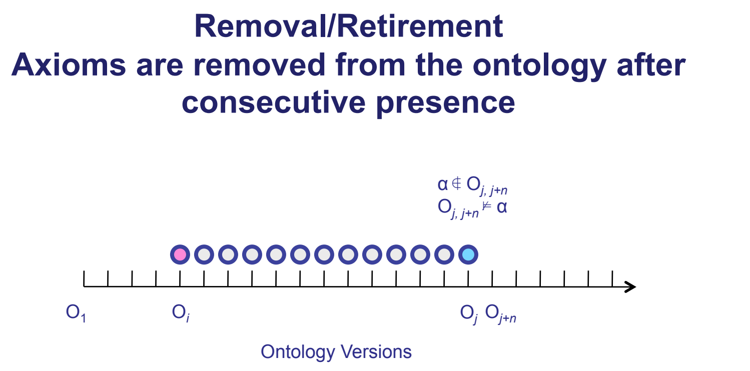

Removal/Retirement
Axioms are removed from the ontology after consecutive presence
$\alpha \notin O_{j,\,j+n}$
$O_{j,\,j+n} \nvDash \alpha$
O_1
O_i
O_j O_{j+n}
Ontology Versions

Figure 3.3: Removal - Removed Entailment Presence

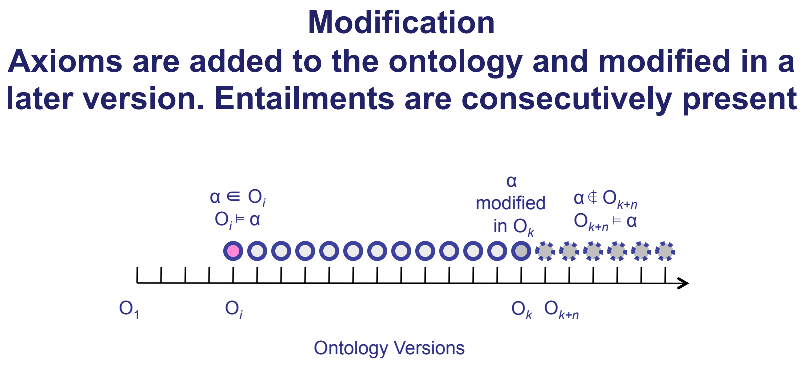


Figure 3.4: Modification - Consecutive Entailment Presence

From these definitions of entailment presence following editing actions, this thesis suggests that the expected and desired logical service that the ontology must comply with can be derived from the axioms ' consecutive entailment presence in the ontology. Knowledge, in the form of axioms, when added to the ontology must remain as logical consequences of the ontology until the entailments are explicitly removed from the ontology. Modifications to existing axioms should not alter the logical consequences of the previously asserted knowledge. Any entailment presence that shows fluctuation in their presence indicates an *undesired* logical behaviour. That is, interrupted entailment presence that results from multiple additions, removals, and modifications of axioms are not in compliance with the expected service of the ontology. This behaviour can be categorised based on the type of interrupted entailment presence; in other words, the effectiveness of the changes. The following definitions and figures show the undesired editing actions and logical consequences of this behaviour.

Definition 3.5. Let O_i and O_j be versions of an ontology O and the axiom α be an effectual addition to O in version O_i, **content regression** of the α entailment in O occurs if α is effectually removed from O in version O_j, and re-enters O as an effectual addition in the later version O_{j+n} where $i < j$ and $n \in \mathbb{N}$.

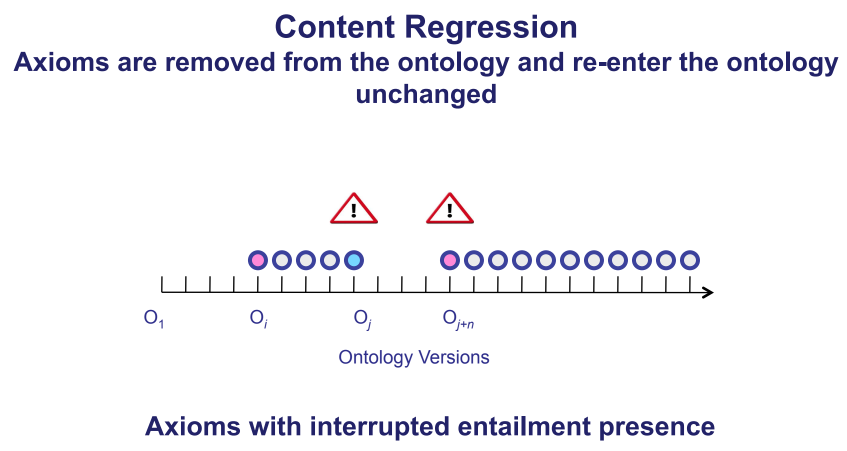

Content Regression
Axioms are removed from the ontology and re-enter the ontology unchanged
O_1
O_i
O_j
O_{j+n}
Ontology Versions
Axioms with interrupted entailment presence

Figure 3.5: Content Regression - Interrupted Entailment Presence

Definition 3.6. Let O_i and O_j be versions of an ontology O and the axiom α be an effectual addition to O in version O_i, **content refactoring** of the α entailment in O occurs if α is ineffectually removed from O in version O_j, and re-enters O as an effectual addition in the later version O_{j+n} where $i < j$ and $n \in \mathbb{N}$.

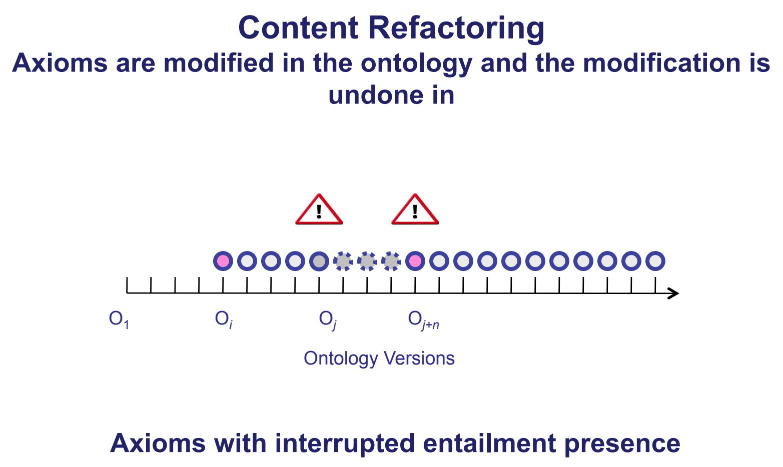

Content Refactoring
Axioms are modified in the ontology and the modification is undone in
O_1
O_i
O_j
O_{j+n}
Ontology Versions
Axioms with interrupted entailment presence

Figure 3.6: Content Refactoring - Interrupted Entailment Presence

Definition 3.7. Let O_i and O_j be versions of an ontology O and the axiom α be an ineffectual addition to O in version O_i, **content redundancy** of the α entailment in O occurs if α is ineffectually added to O in version O_j where $i < j$ and $n \in \mathbb{N}$.

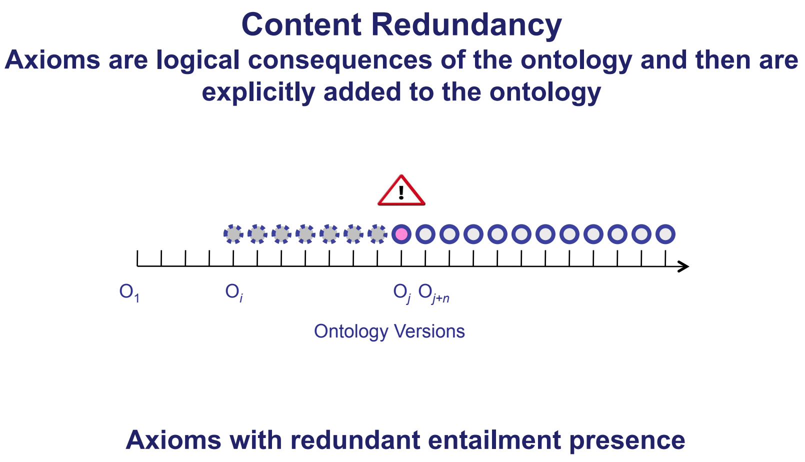

Content Redundancy
Axioms are logical consequences of the ontology and then are explicitly added to the ontology
O1
Oi
Oj Oj+n
Ontology Versions
Axioms with redundant entailment presence

Figure 3.7: Content Redundancy - Interrupted Entailment Presence

These new distinctions of *domain modelling bugs* for redundancy or regression in the entailments indicate an unawareness on the ontology developer's part. This is because the effectuality of the editing actions are not in par with the expected logical behaviour. These pathological editing actions and the unwanted effectuality of these actions are domain modelling bugs that point to suspected missing entailments or unwanted entailments.

3.4 Discussion

This chapter presented the experimental plan, set up and methods that are used for the remainder of this thesis. The data extractions and the data representation structures detailed in this chapter, will allow us to perform SQL-driven data analysis that will paint a statistical portrait of the evolution of the NCIt from 2003 to 2013. This evolution will be analysed with methods that allow for the identification of growth and decline cycles, patterns to these cycles, and the axioms frequency distribution. To understand the logical consequences and the effects to the expected and desired service of the ontology, we also defined expected editing actions and their logical consequences, and expanded the categories of domain modelling bugs to include unwanted and missing entailments resulting from content regression, refactoring, and redundancy. The underlying assumptions regarding the expected editing behaviour are possible to deduce because of the in-depth understanding of the Ontology Life Cycle presented in the preliminaries of this thesis, and are validated against NCIt 's quality assurance and maintenance processes ([61, 65]) which shows this understanding in practice. The next chapter explains in detail the results from these analyses and experiments, and new information and techniques for validating a versioned ontology are proposed.

Chapter 4

Version Analysis for Fault Detection

In the preceding chapters, we have shown that ontology versions are the factual records that capture the released product as it is intended for distribution and use. Although requirements documentation and release reports may be available as part of the ontology 's version release, it is unwise to assume that this documentation is either complete or represents the ontology accurately. In this thesis we claim that an accurate representation of the ontology 's desired services is captured in the ontology 's versions. We have also detailed an experimental plan that will demonstrate this claim by analysing the entailment presence of the asserted axioms in the ontology 's versions. We argue that the persistence of these entailments can point to the desirability of the logical services the ontology must provide. Any interruptions to this service that is outside the 'standard' maintenance process of the ontology can be seen as being out of compliance with the intended service. Each version of the ontology provides a view of the state of the ontology at a point in time, which is the result of the maintenance process that has been applied to deliver the expected service.

In this chapter we apply the experimental plan to the NCIt corpus and provide a study of the evolution and the historical logical footprint of the ontology through out the observed lifetime from 2003 to 2013. Each version is studied on its own and as part of the whole evolution record to observe (i) logical behaviour in the current version, (ii) whether or not this behaviour is inherited from previous versions, (ii) the effectuality of editing actions from version to version (iv) emerging change patterns that are prevalent through out the evolution of the ontology, and (v) the detection and classification of asserted axioms that deviate from the observed entailment lifetime and from the standard editing actions. Verification of the results of this study with the NCIt developers is not part of the scope of this thesis; however, in the last chapter we detail how this confirmation can be conducted and future expansions to this work.

4.1 Assert Axioms Lifetime Profile Report

A cross-sectional study of asserted axioms in each version of the NCIt corpus provides a view of the lifetime record of the ontology. By taking a collective view of the total number of asserted logical axioms in each version of the corpus, we produce a Time Series analysis of the ontology. This Time Series analysis not only shows increases and decreases in the number of asserted axioms in each version of the ontology, but also how many versions in the corpus are involved in these growth and decline cycles. The identification of these cycles can guide observations about growth years and stability periods of the ontology. From this information, we can provide historical context to the editing actions observed for the asserted axioms in each version. We can also provide further comment on the entailment 'stability' of these asserted axioms. We can safely postulate that versions and their corresponding asserted axioms and inferences that are part of these growth cycles may not be as permanent as other axioms that are later introduced to the ontology.

This focus on the stability of asserted axioms and their entailments is further supported by analysing the entailment presence and whether this presence was consecutively observed or interrupted between versions. This analysis is conducted with the methods detailed in the previous chapter, namely, Frequency Distribution analysis. It is from this foundation that we can later detect domain modelling bugs for the axioms that deviate from standard editing actions and their entailment presence.

4.1.1 Assert Axioms Time Series Analysis

The lifetime profile of the NCIt begins with a look at the evolution of the ontology as recorded in each version for the asserted logical axioms. The results from this Time Series analysis show that the growth of the NCIt ontology from September 2003 to May 2013 can be characterised by an upward linear trend, where the ontology grows at a rate of 1.01% per version with a mean difference of 798 added axioms in each new subsequent version. In the Table 4.1, we show that three major periods of growth were identified in the NCIt evolution. These observations are: (1) from versions O_4 to O_5 with 9.65% growth, (2) from versions O_5 to O_6 with 23.45% growth, and (3) from versions O_9 to O_{10} with 9.83% growth [1]. The table also shows two decline periods where a larger than average number of axioms were removed from the ontology. These periods correspond to a major decline from versions (4) O_{16} to O_{17} with a 28.50% decline, and a minor decline from versions (5) O_{26} to O_{27} with a decline of 1.15%.

When analysing the results from the Time Series analysis as a linear graph, Figures 4.1 and 4.2, we see that between September 2003 (with version O_1 being the first OWL version released for the NCIt and the first version in our corpus) and March 2006 (version O_{27}) there is an active period of change in terms of additions and removals of asserted axioms. In these 'formative years' of the NCIt ontology, the asserted content fluctuates greatly from the mean content editing patterns observed in later years. During this period the mean growth rate is of 2.13% with an observed difference of 1125 added axioms per version.

From these results and the identification of the ontology 's 'formative' period, we can apply historical context to individual asserted axioms that seem to deviate from the expected entailment pattern. In addition, we can expect that any interruptions to the asserted axioms ' entailment presence is more likely to take place during this period of major rework in the ontology.

Table 4.1: Time Series Analysis for NCIt from Sep 2003 to May 2013

Observation Num.	Versions O_i	Change Rate
1	O_4 to O_5	9.65%
2	O_5 to O_6	23.45%
3	O_9 to O_{10}	9.83%
4	O_{16} to O_{17}	-28.50%
5	O_{26} to O_{27}	-1.15%

[1]For a mapping between the version names used in this thesis and the NCIt file names, refer to Appendix A

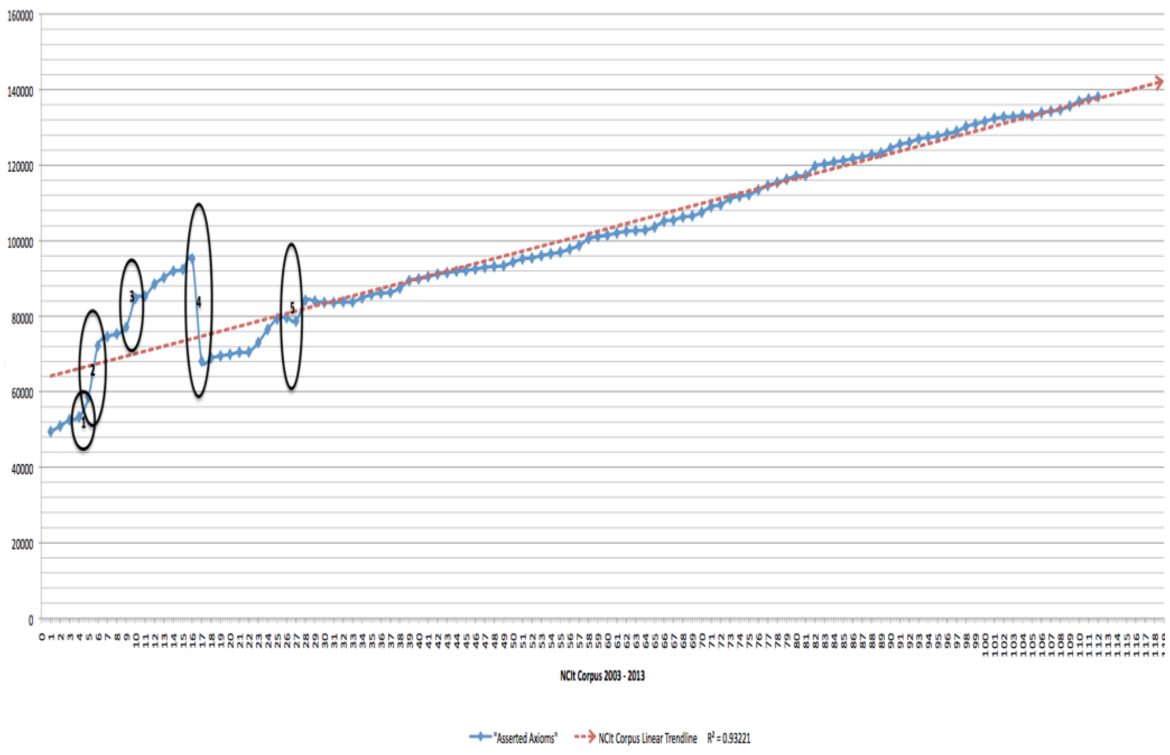


Figure 4.1: Asserted Axioms Count - Time Series (x-axis: NCIt version, y-axis: number of axioms).

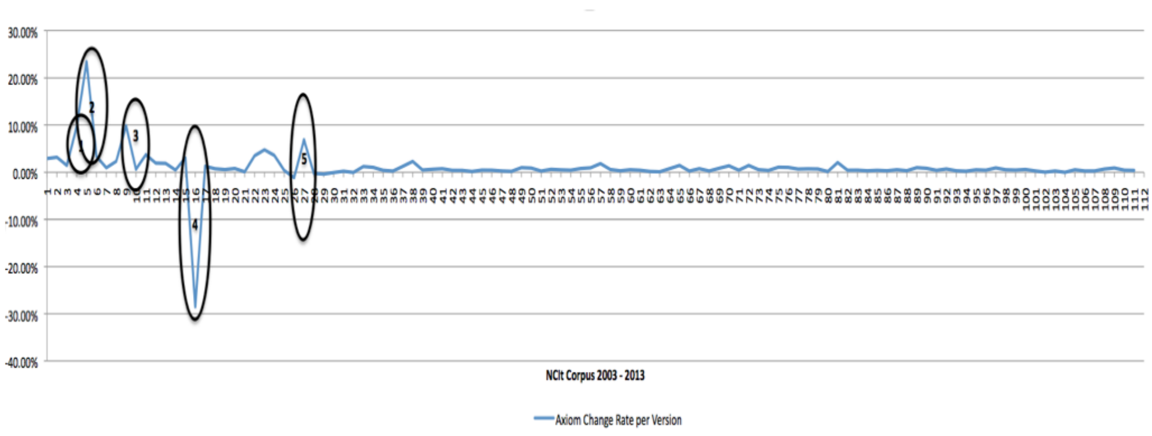


Figure 4.2: Mean Change Rates (x-axis: NCIt version, y-axis: mean change rate).

4.1.2 Asserted Axioms Frequency Distribution Analysis

Frequency Distribution analysis looks at the frequency, or count, of the number of versions each asserted axiom is present in. This measurement indicates the presence consistency, or stability, of each asserted axiom in the ontology. To further explore the nature of this consistency, we also look at the type of presence for each axiom in the ontology. That is, whether the axiom is present in consecutive versions or whether it enters and leaves the ontology to later re-enter in a different version. As seen in Chapter 3, interrupted entailment presence may indicate content regression or refactoring, both of which are anomalies in the editing cycle of axioms. As such, the identification of these asserted axioms is an important step in the experimental plan for detecting domain modelling bugs, and the first step in the detailed study of the entailment profile for these axioms.

The results of the Frequency Distribution analysis revealed that the highest number of versions' frequency is 11, with 20,619 asserted axioms present in 11 versions of the NCIt. When looking at the nature of this presence, the analysis revealed that 20,613 axioms are present in consecutive versions, with 6 asserted axioms appearing in 11 non-consecutive versions. With help from the results of the Time Series analysis, we can place this observation in the context of the ontology's lifetime profile. From a lifetime perspective, we found the distribution of these axioms is primarily concentrated between versions O_6 to O_{16} with 19,384 asserted axioms, between versions O_1 to O_{56} with 607 asserted axioms, and between versions O_{102} to O_{112} with 488 asserted axioms. This observation aligns with our previous assessment about the 'formative' period of the NCIt. The majority of asserted axioms with frequency 11 are present in this period, which indicates fluctuating content between the years 2003 and 2006.

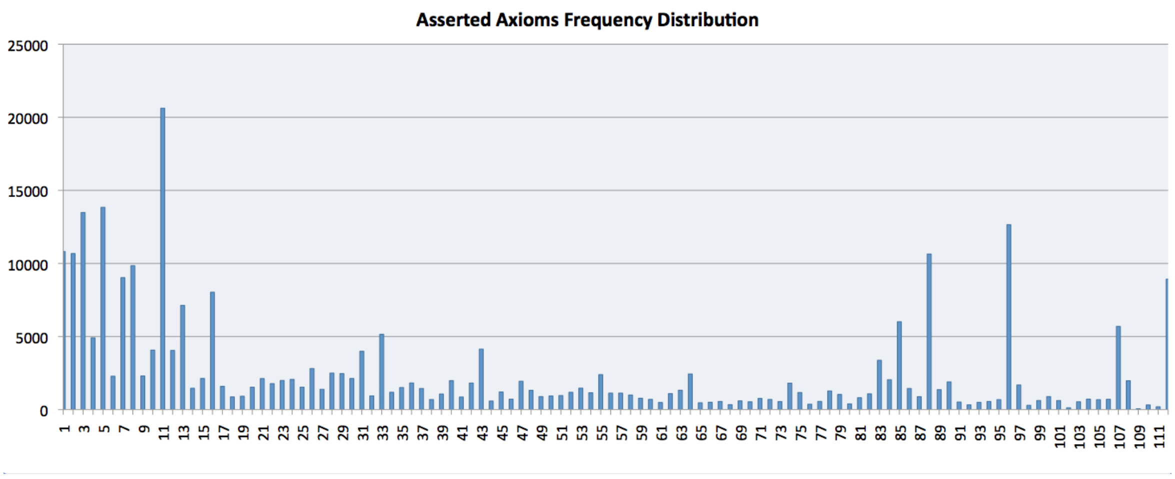


Figure 4.3: Distribution of asserted axioms based on the number of versions they are present in (x-axis: frequency, y-axis: number of asserted axioms).

The next two highest frequencies are 5 and 3 with 13,840 and 13,486 asserted axioms respectively. For frequency 5, 13,840 asserted axioms appear in consecutive versions. The concentration of these consecutive appearances is highest from versions O_1 to O_5 with 9,567 asserted axioms, and from O_{12} to O_{16} with 2,422 asserted axioms. For the next highest frequency of 3, we found that 13,485 axioms are found in consecutive versions with the highest concentration distributed between versions O_1 to O_{11} with a total of 9,595 asserted axioms. Only one axiom for each one of these frequencies has interrupted presence in the NCIt.

Table 4.2: Axioms Count for Frequency Distribution Trends

Frequency	Total Asserted Axioms	Consecutive	Non-Consecutive
11	20,619	20,613	6
5	13,840	13,839	1
3	13,486	13,485	1
96	12,653	12,632	21
1	10,815	-	-
2	10,676	10,670	6
88	10,645	10,639	6
8	9,847	9,844	3
7	9,028	9,020	8
112	8,922	8,922	-

As seen in Table 4.2, there is a large number of asserted axioms that appear only once in the NCIt corpus. These short lived asserted axioms amount to 10,815 total axioms that occur in 101 versions out of the 112 studied versions of the NCIt. These logical axioms with single presence indicate that they were possibly wrongly added to the ontology and this error was immediately rectified in the next version. Although these deletions can be seen as a standard editing action, it is important to analyse the effectiveness of the deletions. An effectual deletion is more likely to indicate a deliberate action by the developers to explicitly remove content from the ontology. If the entailment persists even after the deletion, the deletion was ineffectual, then this ineffectiveness goes against the desire to remove the knowledge from the ontology, and against the ontology 's expected logical service. From the ontology 's lifetime perspective, Figure 4.4 shows that axioms with single presence in the corpus mainly occur from versions O_1 to O_{34} which maps against the 5 periods of major fluctuation identified in Table 4.1. This is further confirmation that the NCIt evolution between September 2003 to March 2006 points to an active changing period where many changes to the content do not remain in the ontology as the ontology matures.

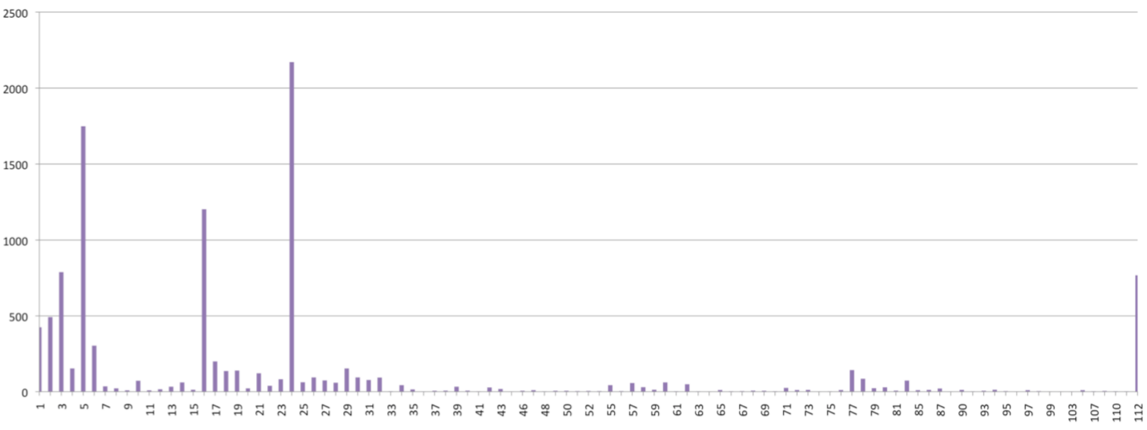


Figure 4.4: Asserted Axioms with Single Presence (x-axis: NCIt version, y-axis: total number of axioms per version).

In contrast to the lowest frequency of 1, we notice that in the top ten frequencies there are three high frequency distributions observed in the NCIt corpus. These frequencies correspond to: (1) frequency 96 with 12,653 asserted axioms, (2) frequency 88 with 10,645 asserted axioms, and (3) frequency 112 with 8,922 asserted axioms. The analysis of these findings against the NCIt lifetime shows that generally the first presence for these axioms occurs between versions O_1 and O_{25}. These high frequencies indicate that out of the total number of 282,152 asserted axioms in the NCIt corpus, 32,220 asserted axioms, or 11.42% of the total number of asserted axioms analysed, stay in more than 88 versions with approximately 99% of these axioms having consecutive presence. Furthermore, 8,922 asserted axioms, or 3.16% of the analysed ontology, entered the ontology in O_1 and have never being modified or deleted. Whether this represents a 'core' set of knowledge, or whether these asserted axioms are 'dead code' that has been deliberately abandoned or simply forgotten, cannot be confirmed from the statistical analysis alone. Content and domain analysis, with the collaboration of the National Cancer Institute, is necessary to establish the nature of these axioms.

The remaining frequencies of 2 with 10,676 asserted axioms, 8 with 9,847 asserted axioms, and 7 with 9,028 asserted axioms are evenly scattered through the NCIt. This is especially the case for frequency 2 where axioms appear for two version only from the first version until the last analysed version. Similarly for frequency 8, we find an even distribution of these asserted axioms across the corpus until version O_{103}. For frequency 7, a high concentration of these asserted axioms is observed in the first 10 versions of the corpus.

The information this analysis provides, regarding the nature of the asserted axioms presence through out the NCIt, allows us to identify those axioms with non-consecutive presence in the NCIt versions. From this analysis, we identify a set of 52 asserted axioms (see Table 4.2) that need further analysis of the entailment presence from an inference record point of view. We can speculate that in addition to the pathological editing actions these asserted axioms have, their logical consequences may also indicate anomalies in their entailment presence. To confirm this claim, we study the effectuality of the editing actions, and classify any undesired logical behaviour as domain modelling bugs based on the definitions presented in Chapter 3.

It is also of interest to this thesis to further analyse the 10,815 asserted axioms that are present in the ontology for only one version. As previously discussed, the effectuality of the deletions for these asserted axioms will point to whether there are desired or undesired logical consequences from the deletions.

4.2 Fault Detection for Domain Modelling Bugs

4.2.1 Defining Domain Modelling Bugs Sequences

When analysing the editing actions of asserted axioms in a versioned ontology, it is important to look at the action, the effectuality of the action, and whether this action is part of a larger set of editing actions. This provides a clear editing profile for each asserted axiom from which we can further analyse the logical consequences of these actions. Asserted axioms that deviate from what is considered standard editing sequence and behaviour in a versioned ontology, must be identified and studied for the possibility of logical faults being introduced to the ontology. Ineffectual editing actions can result in the introduction of undesired or wrong entailments in later versions of the ontology. Unlike other logical faults in OWL ontologies, wrong entailments indicate a failure to adhere to the expectation of the desired logical service, see Chapter 2. These logical faults indicate human error with regards to the understanding of the domain and the logical consequences of the asserted knowledge in the ontology. Presently, automatic detection of such errors is extremely difficult due to the lack of quantification techniques that can be applied to test the intentions and consequences of developers actions. When looking at editing actions in a vacuum, separate from the lifetime profile of the ontology, there is simply no data that can help judge whether a set of editing actions and the consequences of these actions comply with the expected service the ontology must provide. It is with the inclusion of the historical record of the ontology assertions and entailments, that we can begin to suggest or indicate whether a particular editing action has the desired effect to the content and logical consequences of the ontology.

So far we have focused on the lifetime profile of the ontology, the frequency and type of presence of the asserted axioms. We have also identified asserted axioms that have non-consecutive presence, which may indicate the introduction of domain modelling bugs to the ontology. We now shift from this ontology 'as-a-whole' perspective to an individual axiom perspective, with the aim of producing an entailment lifetime profile for each of the 52 asserted axioms identified in the previous section. In order to produce the entailment profiles, we not only need to look at the frequency of presence and the effectuality of the editing actions, but also at the combinations, or sequences, of these actions. This is important because we are looking at the combined historical record of editing actions for asserted axioms. This historical view of previous editing events can provide information about the desirability of the action, and suggest why a future editing action took place in a later version. For instance, the addition of an asserted axiom to the ontology in version O_i may be after the removal of this axiom in a previous version. With this information, we can see that this addition may indicate content regression, or it may be an attempt to 'fix' the erroneous deletion. The addition is no longer a simple addition to the ontology, it now has historical editing context that indicates the introduction or the attempt to fix a logical fault in the ontology. With this in mind, it is important to establish precise definitions for anomalies in editing actions sequences and the entailment fluctuations we are trying to isolate.

Definition 4.1. Let O_i, O_j, O_k, O_l, be versions of an ontology O where $i < j < k < l$. An axiom α has a **fault indicating set of changes** in O for versions O_i, O_j, O_k, O_l, denoted as $\text{FiSoC}((i, j), (k, l))$, if $\alpha \in \text{FiSoC}((i, j), (k, l)) := \text{EffAdd}(O_i, O_j) \cap \text{EffRem}(O_k, O_l)$

This sequence of changes indicates an explicit removal of content from the ontology. That is, the asserted axiom α was effectually added to the ontology in version O_j and removed effectually from the ontology in version O_l. Although this editing behaviour can be expected in a maintained ontology, we stress that it is important to document this change as it may indicate the introduction of a domain logical bug. If the axiom α in a series of versions of ontology O has a *FiSoC* sequence of editing actions, either the entailment $O_j \models \alpha$, or the non-entailment $O_l \not\models \alpha$ is a domain modelling bug because it indicates a change in the asserted knowledge of the ontology. It is very important to note that the identification of the domain modelling bug for α does not determine which of these two actions (either the addition or the deletion) introduces the domain modelling bug. Instead the fault indicating sequence tells us that *one* of the changes introduces a change to the domain. If any subsequent findings of the same type are found for α in future versions, say for versions O_m, O_n, O_o, O_p, in O and $\alpha \in \text{FiSoC}((m, n), (o, p))$, then this set of *FiSoCs* with its newly introduced effectual addition in O_n is evidence of a **content regression** (refer to Definition 3.5).

The case when a sequence of changes includes an ineffectual removal, $\alpha \in \mathit{IneffRem}(O_i, O_j)$, we suggest this editing action should also be identified as a possible problematic modelling action. Although the logical functionality of the ontology does not change by an ineffectual removal, such a sequence of changes (e.g effectual addition, ineffectual removal) is indicative of **refactoring** (see Definition 3.6). Of course, if the effectual addition is a domain modelling bug (this addition was not intended to take place), then the ineffectual removal from O_i to O_j would be a failed attempt to remove the bug. When examining the possible sequences of changes for ineffectual additions and removals, we can conclude that an iterated pattern of ineffectual changes is problematic, and must be flagged for attention. Specifically, even when a set of changes of the type $\alpha \in \text{EffAdd}(O_i, O_j) \cap \text{IneffRem}(O_k, O_l)$ is identified as refactoring, a subsequent ineffectual addition in later versions of the ontology, $\alpha \in \text{IneffAdd}(O_m, O_n)$ where $m < n$, would indicate a sort of content *thrashing*. Meaning, if the original refactoring was correct, then ‘refactoring back’ is a mistake (and if the ‘refactoring back’ is correct, then the original refactoring is a mistake).

Based on this insight, we propose the following formal definitions for a sequence of changes that include ineffectual editing actions.

Definition 4.2. Let O_i, O_j, O_k, O_l, be versions of an ontology O where $i < j < k < l$. An axiom α has a **fault suggesting set of changes** in O for versions O_i, O_j, O_k, O_l, denoted as $\text{FSSoC}((i,j),(k,l))$, if α has sequence of changes such that:

- $\alpha \in \text{F1SSoC}((i,j),(k,l)) := \text{EffAdd}(O_i,O_j) \cap \text{IneffRem}(O_k,O_l)$
- $\alpha \in \text{F2SSoC}((i,j),(k,l)) := \text{IneffAdd}(O_i,O_j) \cap \text{IneffRem}(O_k,O_l)$
- $\alpha \in \text{F3SSoC}((i,j),(k,l)) := \text{IneffRem}(O_i,O_j) \cap \text{IneffAdd}(O_k,O_l)$

When considering the historical record of the editing actions for the asserted axiom α, we can see that these sequences of changes may appear as a combinations of *FSSoC* and *FiSoC* sets. For instance, consider the case where an ontology O with versions $O_i, O_j, O_k, O_l, O_m, O_n$, and an asserted axiom α with the following sequence of changes $\alpha \in \text{EffAdd}(O_i,O_j) \cap \text{IneffRem}(O_k,O_l) \cap \text{IneffAdd}(O_m,O_n) \cap \text{EffRem}(O_o,O_p)$. From what follows, we can say that α has an indicative fault in the sequence $\text{EffAdd}(O_i,O_j)$, $\text{EffRem}(O_o,O_p)$ and two suggestive faults in the sequence, $\text{EffAdd}(O_i,O_j)$, $\text{IneffRem}(O_k,O_l)$ and $\text{IneffRem}(O_k,O_l)$, $\text{IneffAdd}(O_m,O_n)$, where the latter two are subsumed by the former. For this reason it is imperative to study the sequence ’s combinations and articulate the effects to the content of the ontology and the impact to the expected logical service.

The analysis of axiomatic editing actions and sequences of these actions, the effectuality of these actions, and context from the entailment presence analysis are key for fault detection techniques in ontologies. This is not only because these retrospective analyses aid the understanding of evolution of the NCIt, but because these analyses and entailment profiles reports allow us to notify and warn the user of possible faulty axioms in the ontology that deviate from the expected logical service in current and future versions. We have demonstrated that this can be completely achieved by applying this thesis ’s methods and definitions to the ontology ’s versions alone, without requiring documentation about requirements, developers intentions, and any other sources outside the ontology files.

In the next session, we present the results of the analyses conducted in the NCIt corpus and evaluate these results to provide a concise report of the logical faults that have been introduced to the NCIt during its lifetime. These diachronic results support the entailment profiles for the 52 asserted axioms with interrupted entailment presence and indicative of domain modelling bugs through out their presence in the NCIt. It is important to note at this point, that although we take into account minor corrections and additions to the ontology, we assume that any major fluctuations in the ontology 's content and erratic editing actions to the asserted knowledge (outside the ontology 's formative period) may indicate that the ontology is in a pathological state. If this is the case, it is difficult to speculate on the expected service and function of the ontology from version analysis alone. As seen in the NCIt 's lifetime analysis report, this is not the case for the NCIt ontology. The NCIt ontology is a 'mature' ontology that has shown a steady evolution for the last 7 years of the analysed corpus.

4.2.2 Domain Modelling Bugs Report for the NCIt Ontology

In the Experimental Plan detailed in Section 3.3, we set up to test fault detection of domain modelling bugs based on the following driving experiments/analyses:

- NCIt 's Lifetime Profile report from 2003 to 2013 (Section 4.1.1)
- Frequency Distribution Analysis for the NCIt 's asserted axioms (Section 4.1.2)
- Evaluation of the effectiveness of the editing actions observed for the NCIt 's asserted axioms with interrupted entailment presence (Section 4.2.1)

From these analyses, we have identified 52 asserted axioms with interrupted entailment presence in the top ten frequency distribution results that need further inspection to isolate and categorise the domain modelling bugs introduced to the NCIt by these axioms. In this last set of results, we present this further effectuality and sequence of change analysis, domain modelling bugs classification, and entailment profiles for these 52 troublesome asserted axioms.

We found that 21 out of the 52 asserted axioms have logical bugs of type FiSoC. Ten of these identified FiSoC domain modelling bugs appear in a sequence of two FiSoC sets; that is, FiSoC$((i,j),(k,l))\cap$FiSoC$((m,n),(o,p))$ where $O_i, O_j, O_k, O_l, O_m, O_n, O_o, O_p$ are versions of the ontology O and $i<j<k<l<m<n<o<p$. When analysing the justifications for these 10 axioms (see Section 2.2.1 for the definition of justifications), we found that the entailments for 7 axioms are a direct result of their assertion in the ontology and not through any other distinct asserted axioms ' logical consequences in that version. The three remaining axioms of type FiSoC have

justifications that include the axiom assertions and other distinct asserted axioms that result in their entailment. For the remaining 11 asserted axioms out of the 21 axioms of type FiSoC, there are 6 axioms that have a single FiSoC sequence followed by an effectual addition where the added effectual assertion holds until version O_{112}. The remaining 5 axioms contain two FiSoC sequences followed by an effectual addition with the assertion holding until version O_{112}.

The sequence of change combinations observed for these FiSoC sets, namely, a FiSoC set followed by another FiSoC set and/or by additional effectual additions, is consistent with the definition for content regressions (see Definition 3.5) due to the presence of one or more effectual additions following the effectual removals. Table 4.3 defines formally the sequences of FiSoC sets that correspond to content regression domain modelling bugs.

Table 4.3: FiSoC sequence types indicating **content regression**:

Type	Structure
1	$\mathit{FiSoC}((O_i, O_j), (O_k, O_l)) \cap \mathit{FiSoC}((O_m, O_n), (O_o, O_p)$
2	$\mathit{FiSoC}((O_i, O_j), (O_k, O_l)) \cap \mathit{FiSoC}((O_m, O_n), (O_o, O_p)) \cap \text{EffAdd}(O_r, O_s)$
3	$\mathit{FiSoC}((O_i, O_j), (O_k, O_l)) \cap \text{EffAdd}(O_m, O_n)$

The presence of FiSoC sequences were also found in sequences that either subsumes, or are in combination with other domain modelling bugs. For instance, axiom `Phytotherapy_or_Herbalism` $\sqsubseteq$ `Complementary_and_Alternative_Medical_Therapy` with axiom id α_{165492}, has a sequence of FiSoC (that is, $\text{EffAdd}(O_{23}, O_{24}) \cap \text{EffRem}(O_{29}, O_{30})$), followed by a sequence of the type *F2SSoC* (sequence, $\text{IneffAdd}(O_{36}, O_{37}) \cap \text{IneffRem}(O_{37}, O_{38})$). Although this sequence does not indicate content regression (due to the absence of an effectual removal in the last set of changes), it does, however, indicate content redundancy introduced in version O_{37} with the ineffectual addition and the ineffectual deletion from O_{37} to O_{38}. As seen in this example, editing changes sequences can occur in much more complicated sets than previously thought, indicating very peculiar editing patterns for axioms identified as domain modelling bugs.

In the case where *FiSoC* subsumes other logical bugs, we find that 5 axioms with a *FiSoC* sequence subsuming a *F1SSoC* followed by a *F3SSoC* sequence. These five axioms (recorded in the corpus as $\alpha_{103206}, \alpha_{105069}, \alpha_{6858}, \alpha_{7229}, and \alpha_{22465}$) follow the form of $\alpha \in \text{EffAdd}(O_i, O_j) \cap \text{IneffRem}(O_k, O_l) \cap \text{IneffAdd}(O_m, O_n) \cap \text{EffRem}(O_o, O_p)$, where α represents each of the 5 identified axioms in O, $O_i, O_j, O_k, O_l, O_m, O_o, O_p$ are versions in O, and $i < j < k < l < m < n < o < p$. The analysis of the justifications for these five axioms show that for each of these axioms the entailments are a result of the assertion and the logical consequence of other distinct axioms in those versions. This resulting entailment profile indicates that the developer may not be aware of the consistent entailment (outside the entailment from the assertions) of these five axioms. The suggestiveness of this observation is strongly supported by the evidence of an ineffectual removal in version O_l, and content redundancy via the ineffectual addition in O_n. It could be argued that these axioms are finally 'fixed' with the effectual removal in version O_p where the assertion and the justifications for the entailment are removed.

In terms of categorising this newly discovered combination of FiSoC sequence of changes, we find this sequence is consistent with the definition for content redundancy. This sequence together with the previous three sequences identified in the Table 4.3, show that content redundancy and regression can be introduced to an ontology in different versions. This again shows the importance of tracking editing actions in a versioned ontology in order to identify potential domain modelling bugs present in the current version of the ontology.

Table 4.4: FiSoC sequence types indicating **content redundancy**:

Type	Structure
4	$\mathit{FiSoC}((O_i,O_j),(O_s,O_t)) \sqsubset (\mathit{F1SSoC}((O_k,O_l),(O_m,O_n)) \cap \mathit{F3SSoC}((O_o,O_p),(O_q,O_r)))$

The next group of logical bugs found in the analysis focuses on unions and subsumptions sequences involving *F1SSoC* fault. In the first set of results for *F1SSoC*, there are 3 axioms of the form $\text{EffAdd}(O_i,O_j) \cap \text{IneffRem}(O_k,O_l) \cap \text{IneffAdd}(O_m,O_n)$, that is, a *F1SSoC* sequence followed by a *F3SSoC* sequence. Axioms identified in the corpus as $\alpha_{3241}, \alpha_{12085}, \alpha_{42533}$ enter the ontology as effectual additions and are followed by an ineffectual removal and addition. The entailment profile for these axioms show the entailments are a result of the assertions and the logical consequences of other distinct axioms. Like in FiSoC sequence Type 4 in Table 4.4, the sequence of changes indicate a lack of awareness of the asserted axioms 's entailments in the ontology through out all of the versions it is present in. The first *F1SSoC* sequence indicates content refactoring where the asserted axiom is modified yet the entailment is kept as a consequence of the ontology. The last ineffectual addition indicates content redundancy since the axiom is added as an element of the ontology when it is in fact already entailed by the ontology.

Another combination of domain modelling bugs found in the study is exemplified by the axiom with id α_{105423}. This axiom presents a *F1SSoC* Type 1 sequence followed by an additional ineffectual removal, formally $\alpha_{105423} \in (\text{EffAdd}(O_6, O_7) \cap \text{IneffRem}(O_{16}, O_{17}) \cap \text{IneffAdd}(O_{23}, O_{24}) \cap \text{IneffRem}(O_{109}, O_{110}))$. This axiom 's pattern of editing actions show a *F1SSoC* bug, followed by a *F3SSoC* sequence, and ending with a *F2SSoC* bug sequence. This structure indicates that content refactoring was introduced in version O_{16}, content redundancy in version O_{24}, and finally followed by another refactoring event in version O_{109}. The entailment profile for α_{105423} shows the assertions take place in versions $O_{7..16}$ and in versions $O_{24..109}$. There are additional inferences for this axiom, not resulting from the assertions, from the versions $O_{7..15}$, $O_{17..57}$, and $O_{60..109}$. The axiom is not entailed by the ontology for two versions, O_{58} and O_{59}, and from version O_{109}. This shows the justifications for the axiom are removed in these versions. Based on the ineffectual removal and addition in versions O_{17} and O_{24} respectively, it is reasonable to suggest that the developer is not aware of the entailment of the axiom by the ontology, and of it's removal from versions 58 to 59. In addition, this editing pattern and the entailment profile also indicate that the developer may not be aware that final removal is an indirect result of the removal of its justifications and not of the developer 's direct action to remove the axiom. The domain modelling bug sequences that indicate content refactoring and redundancy can be found in Table 4.5

Table 4.5: F1SSoC sequence types indicating **content refactoring** and **redundancy**:

Type	Structure
1	$F1SSoC((O_i, O_j), (O_k, O_l)) \cap F3SSoC((O_m, O_n), (O_o, O_p))$
2	$F1SSoC((O_i, O_j), (O_k, O_l)) \cap F3SSoC((O_m, O_n), (O_o, O_p)) \cap F2SSoC((O_q, O_r), (O_s, O_t))$

The next set of results indicate refactoring and thrashing of axioms. We found that 9 axioms, represented in the corpus with axiom ids: $\alpha_{106537}, \alpha_{106569}, \alpha_{106878}, \alpha_{107407}$, $\alpha_{107860}, \alpha_{107952}, \alpha_{108468}, \alpha_{111380}, \alpha_{114549}$, have *F1SSoC* sequence of changes followed by a sequence of type *F2SSoC*. The sequences of changes for all 9 axioms have these editing actions involved in these versions: $\text{EffAdd}(O_8, O_9) \cap \text{IneffRem}(O_{16}, O_{17}) \cap \text{IneffAdd}(O_{24}, O_{25})$. This set of actions shows that content refactoring takes place in version O_{16}, where a failed attempt to remove the assertion occurs. There is also presence of content thrashing, or refactoring-back, taking place with the ineffectual addition for version O_{25}. The entailment profiles for these axioms indicate the assertions from versions O_9 consecutively to version O_{16}, and from versions O_{25} consecutively to the last version in the corpus O_{112}. The justifications for these 9 axioms show inferences via assertions and logical consequences of ontology for versions $O_{9..15}$, $O_{17..57}$, and $O_{60..112}$.

A final sequence of faults for the domain modelling bug of type *F1SSoC* involves a *F1SSoC* sequence subsuming another *F1SSoC* sequence, followed by a *F3SSoC* and by *F2SSoC* sequences. We found 3 axioms, identified as α_{127241}, α_{9761}, α_{157661}, showing this type of editing pattern. The editing structure corresponds to $\alpha \in \text{EffAdd}(O_i, O_j)$ $\cap \text{IneffRem}(O_k, O_l) \cap \text{IneffAdd}(O_m, O_n) \cap \text{IneffRem}(O_o, O_p)$ where axiom refactoring, or a possible failed attempt of deletion, takes place in O_k. We also find content thrashing and redundancy for version O_n, and a final refactoring for version O_o. The entailment profiles show that α_{127241}, α_{127241} are elements of the ontology for versions O_{16} and O_{21}. These axioms are logical results of the ontology for versions $O_{17..112}$. In the case of the axiom α_{9761}, there are fluctuating inferences, which indicate that the justifications for α_{9761} are removed and reintroduced through out the lifetime of axiom. Finally, the entailment profile for axiom α_{157661} shows that the assertions and logical consequences correspond from versions O_{20} to O_{23}, and in version O_{45}.

Table 4.6: F1SSoC sequence types indicating **content refactoring** and **thrashing**:

Type	Structure
3	$F1SSoC((O_i,O_j),(O_k,O_l)) \cap F2SSoC((O_m,O_n),(O_o,O_p))$
4	$F1SSoC((O_i,O_j),(O_k,O_l)) \sqsubset (F1SSoC((O_m,O_n),(O_o,O_p)) \cap F3SSoC((O_q,O_r),(O_s,O_t))$ $\cap F2SSoC((O_u,O_v),(O_w,O_x)))$

For the logical type *F2SSoC*, we found a single editing sequence pattern where *F2SSoC* subsumes a *F3SSoC* fault. This fault sequence is found for the axiom `Melanocytic_Nevus` $\sqsubseteq$ `Melanocytic_Neoplasm` with axiom id α_{110594}. The editing pattern for this axiom shows $\alpha_{110594} \in \text{IneffAdd}(O_9, O_{10}) \cap \text{IneffRem}(O_{18}, O_{19}) \cap$ $\text{IneffAdd}(O_{30}, O_{31}) \cap \text{IneffRem}(O_{30}, O_{31})$. All of the observed editing actions for this axiom are ineffectual in their nature (there are other axioms in the ontology that have α_{110594} as their logical consequence). Any attempt for explicit addition to the ontology is redundant and shows a lack of awareness of the inference of this axiom by the ontology. Specifically, we find that α_{110594} is an element from versions $O_{10..19}$ and versions O_{31}. This axiom is a logical consequence of the ontology for versions $O_{10..19}$, $O_{31..57}$, and versions $O_{60..112}$.

As seen in these results, the sequence of editing actions in the NCIt for axioms with interrupted entailment presence is far more complex than previously predicted. This

Table 4.7: F2SSoC sequence types indicating **content redundancy**:

Type	Structure
1	$F2SSoC((O_i, O_j), (O_k, O_l)) \sqsubset F3SSoC((O_m, O_n), (O_o, O_p))$

observation is strongly correlated with a lack of awareness, on the developers part, of the logical consequences of the ontology and the previous editing actions on these axioms. The introduction of repeated content regressions, redundancies, and refactoring shows that there is wasted effort and lack of understanding of the asserted content. Although there are possible scenarios where the developer may need to introduce assertions even when the inference for that assertion is already present in the ontology (e.g. for publishing the classification hierarchy), the repeated additions and deletions without significant change to the content shows a problematic and faulty editing behaviour.

A further analysis of the results is presented in the last section of this chapter. The following Tables 4.8, 4.9, 4.10, 4.11, and 4.12 help summarise the results presented here. This consolidated view constitutes the entailment profile report for all 52 analysed asserted axioms.

Table 4.8: *FiSoC* Faults with Content Regression - Entailment Profiles

FiSoC Type	α_{id}	O_{id} for Axiom Assertions ($\alpha \in O$)	O_{id} for Axiom Inferences ($O \models \alpha$ and $\alpha \notin O$)
1	57506	4, 7 to 16	4, 7 to 16
	58364	4, 7 to 16	4, 7 to 16
	120551	12, 16	12, 16
	172613	25, 62	25, 62
	172917	25, 62	25, 62
	67505	5, 10 to 16	5, 10 to 16
	48564	1 to 5, 15 to 16	1 to 5, 15 to 16
	210295	40 to 46, 51 to 54	40 to 46, 51 to 54
	50858	2, 18	2, 18
	73441	6 to 11, 79	6 to 11, 79
2	30433	1 to 11, 14 to 47, 89 to 112	1 to 15, 17 to 57, 60 to 74, 89 to 112
3	39267	1, 18 to 112	1, 18 to 112
	68617	5, 18 to 112	5, 18 to 57, 60 to 112
	118516	12 to 73, 79 to 112	12 to 15, 17 to 73, 79 to 112
	119326	12 to 73, 79 to 112	12 to 15, 17 to 73, 79 to 112
	121919	13 to 46, 51 to 112	13 to 15, 17 to 46, 51 to 112
	122832	13 to 46, 51 to 112	13 to 15, 17 to 46, 51 to 112
	8905	1 to 5, 30 to 112	1 to 5, 30 to 112
	125718	15 to 18, 29 to 112	15 to 18, 29 to 112
	125895	15 to 18, 29 to 112	15 to 18, 29 to 112
	162304	23 to 33, 36 to 112	23 to 33, 36 to 112

Table 4.9: *FiSoC* Faults with Content Redundancy - Entailment Profiles

FiSoC Type	α_{id}	O_{id} for Axiom Assertions ($\alpha \in O$)	O_{id} for Axiom Inferences ($O \models \alpha$ and $\alpha \notin O$)
4	103206	7 to 16, 25	7 to 15, 17 to 25
	105069	7 to 16, 25	17 to 25
	6858	1 to 16, 23 to 112	1 to 15, 17 to 102
	7229	1 to 16, 23 to 112	1 to 15, 17 to 102
	22465	1, 45 to 51	1, 45 to 51

Table 4.10: *F1SSoC* Faults with Content Refactoring and Redundancy - Entailment Profiles

F1SSoC Type	α_{id}	O_{id} for Axiom Assertions ($\alpha \in O$)	O_{id} for Axiom Inferences ($O \models \alpha$ and $\alpha \notin O$)
1	3241	1 to 6, 23 to 112	1 to 6, 23 to 113
	12085	1 to 16, 33 to 112	1 to 15, 17 to 112
	42533	1 to 16, 24 to 95	1 to 15, 17 to 57, 60 to 95
2	105423	7 to 16, 24 to 109	7 to 15, 17 to 57, 60 to 109

Table 4.11: *F1SSoC* Faults with Content Refactoring and Thrashing - Entailment Profiles

FiSoC Type	α_{id}	O_{id} for Axiom Assertions ($\alpha \in O$)	O_{id} for Axiom Inferences ($O \models \alpha$ and $\alpha \notin O$)
3	106537	9 to 16, 25 to 112	9 to 15, 17 to 57, 60 to 112
	106569	9 to 16, 25 to 112	9 to 15, 17 to 57, 60 to 112
	106878	9 to 16, 25 to 112	9 to 15, 17 to 57, 60 to 112
	107407	9 to 16, 25 to 112	9 to 15, 17 to 57, 60 to 112
	107860	9 to 16, 25 to 112	9 to 15, 17 to 57, 60 to 112
	107952	9 to 16, 25 to 112	9 to 15, 17 to 57, 60 to 112
	108468	9 to 16, 25 to 112	9 to 15, 17 to 57, 60 to 112
	111380	9 to 16, 25 to 112	9 to 15, 17 to 57, 60 to 112
	114549	9 to 16, 25 to 112	9 to 15, 17 to 57, 60 to 112
4	127241	16, 21	17 to 112
	9761	1 to 16, 24 to 95	1 to 15, 17 to 57, 60 to 95
	157661	20 to 23, 45	20 to 23, 45

Table 4.12: *F2SSoC* Faults with Content Redundancy - Entailment Profiles

FiSoC Type	α_{id}	O_{id} for Axiom Assertions ($\alpha \in O$)	O_{id} for Axiom Inferences ($O \models \alpha$ and $\alpha \notin O$)
1	110594	10 to 19, 31	10 to 19, 31 to 57, 60 to 112

4.2.2.1 Missing Records in the Entailment Profiles Analysis

The results presented above exclude the following axioms with ids: α_{159025}, α_{99659}, α_{153578}, α_{7384}, α_{23034}, α_{44436}, α_{59751}, α_{158782}, where missing entailment history was found in the database tables. The extraction of the entailment records for these axioms does not map with the observed effectual and ineffectual editing actions. For instance, for the axiom α_{44436} with a sequence of changes of $\mathrm{EffAdd}(O_0, O_1) \cap \mathrm{IneffRem}(O_6, O_7) \cap \mathrm{IneffAdd}(O_{15}, O_{16}) \cap \mathrm{EffRem}(O_{16}, O_{17})$, we find its entailment profile show the assertions for versions $O_{1..6}$ and version O_{16} correctly. However, the entailment profile only shows the inference for versions $O_{1..5}$ and not for versions $O_{6..16}$. The inference record in these versions was expected because the database tables containing the editing history (see Schema Representation Figure 3.1) show ineffectual changes for these versions.

These missing inferences are most likely a consequence of an error in the version of the OWL API ([68]) used to compute the inferences of axioms prior to the extraction to the database. We ran the experiment multiple times and every time this problem was present for these axioms. Testing and tracking debugging tools used against this thesis ' JAVA program, and the testing of the execution of the SQL queries gives us confidence that the error originates at the OWL API level and it is outside the methods applied in the experiment.

The entailment profiles produced in this thesis rely entirely on the extraction methods and database representation of these extractions described in Section 3.3.1, and previously applied in the work by ([16, 19]). Each asserted axiom profile is assembled by joining together, via SQL queries, the ontology versions for the assertions and inferences present in the ontology versions ' tables. A confirmation query is ran against the entailment tables to verify the expected assertions and inferences. Although this approach proves effective for the construction of entailment profiles, it relies heavily on data extraction and SQL analysis. In Chapter 5, we provide a logic-based approach for entailment profile extraction that uses the atomic decomposition of the entire NCIt corpus (see ([43])). This updated method is a more robust approach for constructing entailment profiles for axioms, since it allows the extraction of basic logic components, or atoms, of modules for each axiom based on an axiom 's signature and the relationships of this atom with other atoms. This is a straight forward method that will not require external data manipulation in the form of SQL queries for joining the axiom 's assertions and inferences.

4.2.2.2 Entailment Profile for Axioms with Single Presence Analysis

In Frequency Distribution analysis, we found that 10,815 asserted axioms are present in the NCIt corpus only for one version. This introduction of content and immediate removal of content is an interesting observation in the NCIt. Although the precise meaning and reason behind these introductions and removals can only be explained by the NCI developers (since no further editing history can inform the service of this content), we can comment on the logical impact and possible use of this briefly present asserted content. Further analysis of this set shows that 768 axioms were added in version 112 and may continue to be in the ontology for future versions. For the remaining 10,047 axioms, we found these are all effectually removed from the ontology in the next version.

The nature of this single presence is interesting from an diachronic perspective, since these axioms could indicate a particular editing purpose that is not evident from the effectuality analysis. It would be interesting to expand the information for these axioms with context from the domain, log files, and any recorded documentation explaining the changes to the versions. We speculate that these axioms may serve as 'scaffolding', or supporting axioms, that help with the introduction of content to the ontology, and, once this content is part of the ontology, the 'scaffolding' is removed from the ontology.

4.3 Discussion

In this chapter we presented in detail the results for the identification of domain modelling bugs in OWL ontologies based on version analysis. We have shown that by analysing the evolution of the NCIt through Time Series analysis and by identifying axioms that have interrupted entailment presence in the ontology, we can identify axioms that are problematic due to their shifting presence. We proposed indications and suggestions of domain modelling bugs based on the sequence of editing changes and the effectuality of these changes. We defined a core set of editing sequences that indicate content regression for the domain modelling bug of the kind *FiSoC*, and its combinations patterns (see Tables 4.3, and 4.4). We also defined the editing actions that suggest domain modelling bugs for the types *F1SSoC*, *F2SSoC*, *F3SSoC* (see Tables 4.5, 4.6, and 4.7), and how the presence of these editing patterns suggests content redundancy and refactoring. Each one of these findings is further supported by the entailment profiles constructed for each one of the analysed axioms by combining the records of assertions and inferences. This thorough study of entailment history even allowed us to identify errors in the extraction method for inferences where the extracted inference misses entailments that are evident from the editing history. The work presented in this chapter satisfies the experimental plan in Chapter 3, and provides strong evidence that fault detection is possible through the study of the versions in an ontology. In the next chapters, we discuss the implications of these findings for ontology evolution and change management support, the main contributions of this thesis, and for future work, where we propose a logic based approach for the analysis and construction of entailment profiles.

Chapter 5

Conclusion

The work presented in this thesis constitutes a body of evidence that shows that fault detection for OWL ontologies can be expanded to include logical warnings on logical bugs that deviate from the expected entailment behaviour. The set of expectations are grounded in the detailed analysis and inclusion of ontology versions as part of the body of information that should be used when identifying, debugging and repairing ontologies. Ontology versions and the entailment history for its logical axioms indicate the desirability of entailments, the effectiveness of changes, and the sustainability of the edited content throughout the versions of the ontology. We have demonstrated a robust understanding of the Ontology Life Cycle Phases and how, by requiring and using documentation about version releases, design decisions, and building methods, we can provide a wider range of data that is important when evaluating an ontology.

We presented a novel approach for defining domain modelling bugs based on the axioms ' frequency of presence in the ontology, the effectuality of changes, the editing patterns, and successfulness of these changes from the moment the axiom enters the ontology until its last presence. This resulted in a comprehensive set of definitions for indications and suggestions of domain modelling bugs, and a description of the complexity found in the domain modelling bugs ' sequences. From this investigation, we defined a new category of logical faults for *content regression, redundancy*, and *thrashing*. These anomalies were confirmed by the analysed axioms ' entailment profiles, which indicate the assertions and inferences for each version the axioms are present. All of these accomplishments have been tested on one of the most collaborative and mature ontologies, the NCIt. The NCIt ontology has been subjected to studies for content evaluation, evolution, maintenance, and quality assurance since its inception (see [67, 16, 60, 61, 7, 66, 64, 65]). Although these studies provide great insight behind the engineering effort of the NCIt, these fail to unite this knowledge in a practical approach that helps ontology modellers and developers with their day to day job. It is in this area of Ontology Maintenance, where this research makes its major contribution. We have argued and confirmed that in addition to external resources to the ontology (such as documentation about the ontology 's development, maintenance, and quality assurance processes) we must include the ontologies ' versions as part of fault detection and quality assessment techniques. Ontologies are large complex systems that need to be supported and repaired within the context of its evolutionary record. Functional requirements, design patterns, expected services, desired entailments, and core activity should be included in the range of methods for supporting Ontology Evaluation. Although this thesis is the beginning for such an integrated method for fault detection techniques, we believe we have provided sufficient fact-based evidence of the efficiency and necessity of such information within IDEs and debugging tools. Once the ontology 's change history and entailments profile is created, it can be reused with every new version to position the changes within the historical context, and help speculate on the expected logical services the ontology must provide. Based on the methods used in this thesis to calculate expected change and growth rate, we argue that the integration of change and entailment history should not need to be updated in every new release. An ontology 's maintenance process can include, as part of their release cycle, an indication of when the entailment profiles and tracking of previous editing actions must be updated (for instance, every six months depending on the ontology 's lifetime profile). This thesis 's methods and techniques show that it is possible to quantify an ontology 's entailment expectations. We have shown that it is not only possible, but essential to the day to day activities of content authoring and revision.

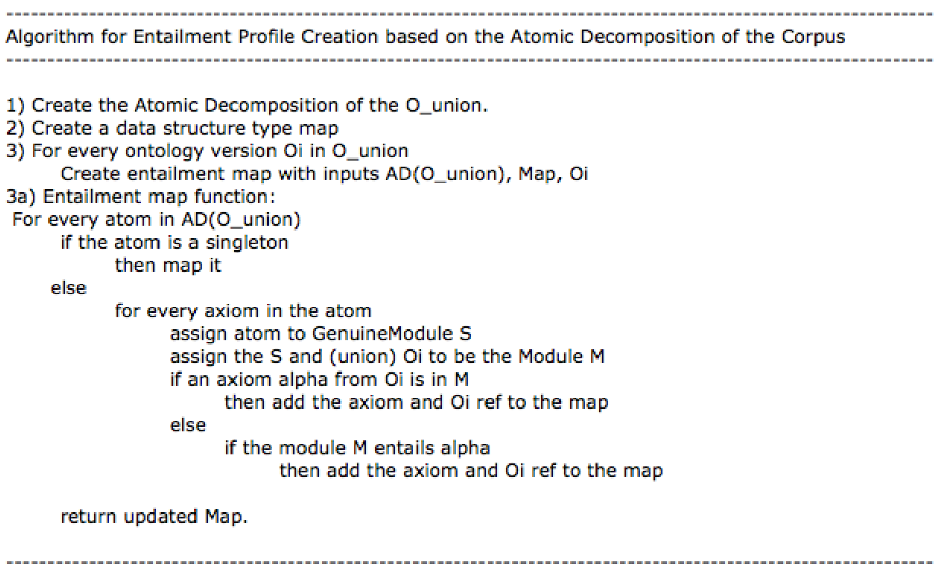
A logic-base approach for entailment analysis, integration with IDEs and other debugging tools, and a full analysis of the NCIt ontology are the next steps and direction of this research. Although the test area selected for this thesis is relatively small in comparison to the size of the NCIt ontology, this thesis and the NCIt Case Study provide strong conceptual methods, and robust analysis techniques for fault detection and warning alerts in OWL ontologies.

5.1 Future Work

The work presented in this thesis provides the foundational underpinnings for the identification of domain modelling bugs that are directly linked to the expectations and desirability of entailments. The research direction that this work must follow is around the area of a logic-based approach for fault detection that can be embedded in existing ontology authoring tools. Although IDEs provide tools for authoring axioms, these tools lack bug detection alerts for content redundancy, meaningless refactoring, or reintroduction of errors. These types of warnings can only be included if records about the expected entailments, and the evolution of the ontology is made available as part of the ontology 's 'compiling' services. Collaboration with the authors and institutions behind the most popular ontology IDEs is required in order to include the ontology 's lifetime report, and asserted axioms ' entailment profiles as information for detecting logical bugs. As seen in this thesis, the detection of domain modelling bugs can be achieved by the methods proposed in this research and should be part of existing fault detection services found in ontology IDEs.

The first step towards a more logic-based approach for this research is in the removal of the dependency on Databases and SQL queries when building entailment profiles. The entailment profiles in this thesis were constructed by analysing the presence of the axioms in the asserted and inferred axioms ' tables. There are more sophisticated methods for calculating the axioms involved in a particular entailment driven by the use of axioms ' signatures. This method, known as the *Atomic Decomposition*, will allow us to build a logical grounded approach for entailment profiles for each identified axiom with an interrupted entailment presence. In the following algorithm we proposed the use of Atomic Decomposition to create the entailment profile for each axiom. This is achieved by examining the asserted axiom 's entailment presence in each of the versions in the corpus. This robust method will eliminate extraction errors from the ontology to the database as we found in the this thesis. (Note: The algorithm refers to the OWL ontology and its versions as the Union Ontology, denoted as O_{union})

Another area of research is in the formal inclusion and representation of functional requirements of the ontology. For instance, we know from the NCIt 's literature that the classification hierarchy is a key requirement and service provided by the ontology. From the analysis of the subsumption hierarchy and the evolution and maturity of the hierarchy, we can define a minimal required hierarchy that must be present in all versions of the ontology. Any axiom that disrupts this expected hierarchy, or set of *Expected Entailments* (denoted as $\mathcal{EE}$), can be highlighted by a warning message for



```
-----------------------------------------------------------------------------------------------------
Algorithm for Entailment Profile Creation based on the Atomic Decomposition of the Corpus
-----------------------------------------------------------------------------------------------------

1) Create the Atomic Decomposition of the O_union.
2) Create a data structure type map
3) For every ontology version Oi in O_union
        Create entailment map with inputs AD(O_union), Map, Oi
3a) Entailment map function:
 For every atom in AD(O_union)
        if the atom is a singleton
                then map it
      else
                for every axiom in the atom
                        assign atom to GenuineModule S
                        assign the S and (union) Oi to be the Module M
                        if an axiom alpha from Oi is in M
                                then add the axiom and Oi ref to the map
                        else
                                if the module M entails alpha
                                        then add the axiom and Oi ref to the map

        return updated Map.

-----------------------------------------------------------------------------------------------------
```

Figure 5.1: Automation of Entailment Profile Creation Algorithm

that axiom. This warning system will ensure that the developer adheres to the required classification hierarchy.

In addition to these future methods and tools, we need to provide more evidence of the methods proposed in this thesis by conducting more in-depth studies in the NCIt corpus to include all asserted axioms in all versions. We also see the need to extend this analysis to other large collaborative ontologies such as the Gene Ontology (GO), and to conduct breath-first studies on ontologies in repositories such as BioPortal[1]. The results from these studies must be validated with collaboration from the ontologies' owners and authors to further confirm and improve the efficiency of detecting domain modelling bugs.

Nevertheless, the contributions from this initial research show potential in the area of Ontology Evaluation. We believe this work makes a strong case for the need of a disciplined approach for documentation and ontology version management since we have proven that these are of great value in fault detection. We have demonstrated that this data does not need to be separated from the ontology building process; it must be integrated and used to verify the logical functionality of the modelled knowledge and prevent future errors when authoring the ontology.

[1]http://bioportal.bioontology.org

Bibliography

[1] M. Horridge, B. Parsia, and U. Sattler, "Lemmas for Justifications in OWL," *Description Logics*, vol. 477, 2009.

[2] A. Kalyanpur, B. Parsia, E. Sirin, and B. Cuenca-Grau, *The Semantic Web: Research and Applications: 3rd European Semantic Web Conference, ESWC 2006, Proceedings*, ch. Repairing Unsatisfiable Concepts in OWL Ontologies, pp. 170–184. Springer Berlin Heidelberg, 2006.

[3] F. Baader, B. Ganter, B. Sertkaya, and U. Sattler, "Completing Description Logic Knowledge Bases Using Formal Concept Analysis," in *IJCAI*, vol. 7, pp. 230–235, 2007.

[4] I. Horrocks, P. F. Patel-Schneider, and F. Van Harmelen, "From SHIQ and RDF to OWL: The making of a web ontology language," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 1, no. 1, pp. 7–26, 2003.

[5] K. Spackman, "SNOMED RT and SNOMED CT. Promise of an International Clinical Terminology," *MD Computing: Computers in Medical Practice*, vol. 17, no. 6, p. 29, 2000.

[6] M. Ashburner, C. A. Ball, and J. A. Blake, "Gene Ontology: tool for the unification of biology," *Nature*, 2000.

[7] S. de Coronado, M. W. Haber, N. Sioutos, M. S. Tuttle, and L. W. Wright, "NCI Thesaurus: Using Science-Based Terminology to Integrate Cancer Research Results," *Studies in Health Technology and Informatics*, vol. 107, no. 1, 2004.

[8] J. L. Mejino Jr and C. Rosse, "Symbolic modelling of structural relationships in the foundational model of anatomy," in *In Proceedings of the KR 2004 Workshop on Formal Biomedical Knowledge Representation*, 2004.

[9] A. J. Duineveld, R. Stoter, M. R. Weiden, B. Kenepa, and V. R. Benjamins, "Wonder Tools? A comparative study of ontological engineering tools," *International Journal of Human-Computer Studies*, vol. 52, no. 6, pp. 1111 – 1133, 2000.

[10] R. S. Gonçalves, B. Parsia, and U. Sattler, "Categorising logical differences between OWL ontologies," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, pp. 1541–1546, ACM, 2011.

[11] A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin, "Finding all justifications of OWL DL Entailments," in *Proceedings of the 6th International Semantic Web Conference*, 2007.

[12] C. Roussey, O. Corcho, and L. M. Vilches-Blázquez, "A catalogue of OWL ontology antipatterns," in *Proceedings of the Fifth International Conference on Knowledge Capture*, pp. 205–206, ACM, 2009.

[13] B. Parsia, E. Sirin, and A. Kalyanpur, "Debugging OWL Ontologies," in *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, (New York, NY, USA), pp. 633–640, ACM, 2005.

[14] E. Simperl, M. Machol, and T. Burger, "Achieving Maturity: the State of Practices in Ontology Engineering in 2009," *International Journal of Computer Science and Applications*, vol. 7, no. 1, pp. 45–65, 2010.

[15] A. Forward and T. C. Lethbridge, "The relevance of software documentation, tools and technologies: a survey," in *Proceedings of the 2002 ACM symposium on Document engineering*, pp. 26–33, ACM, 2002.

[16] R. S. Gonçalves, B. Parsia, and U. Sattler, "Analysing the evolution of the NCI Thesaurus," in *Computer-Based Medical Systems (CBMS), 2011 24th International Symposium on*, pp. 1–6, IEEE, 2011.

[17] J. Malone and R. Stevens, "Measuring the level of activity in community built Bio-Ontologies," *Journal of Biomedical Informatics*, vol. 46, no. 1, pp. 5–14, 2013.

[18] M. Copeland, A. Brown, H. E. Parkinson, R. Stevens, and J. Malone, "The SWO Project: A Case Study for Applying Agile Ontology Engineering Methods for Community Driven Ontologies," in *International Conference on Biomedical Ontology - ICBO*, 2012.

[19] M. Copeland, R. S. Gonçalves, B. Parsia, U. Sattler, and R. Stevens, "Finding fault: detecting issues in a versioned ontology," in *The Semantic Web: ESWC 2013 Satellite Events*, pp. 113–124, Springer, 2013.

[20] T. D. Wang, B. Parsia, and J. Hendler, *The Semantic Web - ISWC 2006: 5th International Semantic Web Conference, Proceedings*, ch. A Survey of the Web Ontology Landscape, pp. 682–694. Springer Berlin Heidelberg, 2006.

[21] O. Corcho, M. Fernandez-Lopez, and A. Gomez-Perez, "Methodologies, tools and languages for building ontologies: where is their meeting point?," *Data and Knowledge Engineering*, vol. 46, no. 1, pp. 41–64, 2003.

[22] A. Gomez-Perez, "Ontological Engineering: A State of the Art," *Expert Update: Knowledge Based Systems and Applied Artificial Intelligence*, vol. 2, pp. 33–43, Autumn 1999.

[23] S. Regoczei and E. P. Plantinga, "Creating the domain of discourse: Ontology and inventory," *International Journal of Man-Machine Studies*, vol. 27, no. 3, pp. 235–250, 1987.

[24] R. Mizoguchi and I. Mitsuru, "Towards Ontology Engineering," Tech. Rep. AI-TR-96-1, The Institute of Scientific and Industrial Research, Osaka University, 1996.

[25] "Oxford Dictionaries." http://oxforddictionaries.com/, June 2011.

[26] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," *International Journal of Human-Computer Studies*, 1995.

[27] R. Mizoguchi, "Knowledge Acquisition and Ontology," *Expert Systems with Applications*, vol. 9, no. 1, pp. 121 – 128, 1993.

[28] M. R. Genesereth and N. J. Nilsson, *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, 1987.

[29] W. N. Borst, "Construction of Engineering Ontologies," tech. rep., University of Twenty - Center for Telematica and Information Technology, Enschede, NL, 1997.

[30] S. Borgo, N. Guarino, and C. Masolo, "Stratified Ontologies: the case of physical objects," in *In Proceedings of the Workshop on Ontological Engineering*, (Budapest), pp. 5–15, 1996.

[31] J. Arpirez, A. Gomez-Perez, A. Lozano, and S. H. Pinto, "ONTO Agent: An ontology-based WWW broker to select ontologies," in *Workshop on Applications of Ontologies and PSMs*, (England), pp. 16–24, 1998.

[32] "Ontology Lifecycle." http://www.uni-koblenz-landau.de/koblenz/fb4/AGStaab/Teaching/SS2005/SemWeb/7-ontology-lifecycle.pdf, June 2011.

[33] "The Development Lifecycle." http://www.cs.man.ac.uk/ stevensr/onto/node13.html, June 2011.

[34] N. Noy and C. D. Hafner, "The State of the Art in Ontology Design - A Survey and Comparative Review," *IA Magazine*, vol. 18, pp. 56–74, Fall 1997.

[35] A. Gomez-Perez and M. Fernandez-Lopez, *Advanced Information and Knowledge Processing*, ch. Ontological Engineering. Springer, 2003.

[36] M. Hepp, P. De Leenheer, A. de de Moor, and Y. Sure, *Ontology management: semantic web, semantic web services, and business applications*, vol. 7. Springer Science & Business Media, 2007.

[37] R. Mizoguchi, "A Step Towards Ontological Engineering," in *Proceedings of the 12th National Conference on AI of JSAI*, pp. 24–31, June 1998.

[38] G. Antoniou, E. Franconi, and F. Harmelen, *Reasoning Web: First International Summer School 2005, Msida, Malta, July 25-29, 2005, Revised Lectures*, ch. Introduction to Semantic Web Ontology Languages, pp. 1–21. Springer Berlin Heidelberg, 2005.

[39] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, eds., *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[40] R. Baader and K. Franz, "Computing the least common subsumer and the most specific concept in the presence of cyclic ALN-concept descriptions," in *In proceedings of KI-98: Advances in Artificial Intelligence, Lectures Notes in Computer Science*, vol. 1504, pp. 129–140, 1998.

[41] B. C. Grau, B. Parsia, E. Sirin, and A. Kalyanpur, "Modularity and web ontologies," in *KR-2006*, pp. 198–209, 2006.

[42] B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler, "Modular reuse of ontologies: Theory and practice," *Journal of Artificial Intelligence Research*, vol. 31, no. 1, pp. 273–318, 2008.

[43] C. Del Vescovo, B. Parsia, U. Sattler, and T. Schneider, "The modular structure of an ontology: Atomic decomposition," in *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, vol. 22, p. 2232, 2011.

[44] "DAM + OIL Website." http://www.daml.org/2001/03/daml+oil-index.html, January 2014.

[45] "W3C Ontology Language Reference." http://www.w3.org/TR/owl-ref/, April 2014.

[46] S. Staab and R. Studer, eds., *Handbook on Ontologies*. Springer, 2004.

[47] N. Matentzoglu, S. Bail, and B. Parsia, *The Semantic Web – ISWC 2013: 12th International Semantic Web Conference, Proceedings, Part I*, ch. A Snapshot of the OWL Web, pp. 331–346. Springer Berlin Heidelberg, 2013.

[48] B. Motik, "On the Properties of Metamodeling in OWL," in *The Semantic Web–ISWC 2005: 6th International Semantic Web Conference, Proceedings*, pp. 548–562, Springer, 2005.

[49] "OWL 2 Web Ontology Language Document Review." http://www.w3.org/2007/OWL/draft/ED-owl2-overview-20090914/, May 2015.

[50] A. A. Kalyanpur, *Debugging and repair of OWL ontologies*. PhD Thesis, The University of Maryland, July 2006.

[51] M. Horridge, B. Parsia, and U. Sattler, "Explaining inconsistencies in OWL ontologies," in *Scalable Uncertainty Management*, pp. 124–137, Springer, 2009.

[52] J. Lehmann and L. Bühmann, "ORE - A tool for repairing and enriching knowledge bases," in *The Semantic Web–ISWC 2010: 9th International Semantic Web Conference, Proceedings*, pp. 177–193, Springer, 2010.

[53] D. Tsarkov and I. Horrocks, “FaCT++ Description Logic Reasoner: System Description,” in *Automated Reasoning*, pp. 292–297, Springer, 2006.

[54] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, “Pellet: A practical OWL-DL reasoner,” *Web Semantics: science, services and agents on the World Wide Web*, vol. 5, no. 2, pp. 51–53, 2007.

[55] T. Redmond and N. Noy, “Computing the changes between ontologies,” in *Joint Workshop on Knowledge Evolution and Ontology Dynamics*, pp. 1–14, 2011.

[56] A. Kalyanpur, B. Parsia, E. Sirin, B. C. Grau, and J. Hendler, “SWOOP: A web ontology editing browser,” *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 4, no. 2, pp. 144–153, 2006.

[57] E. Mikroyannidi, L. Iannone, and R. Stevens, “RIO: The regularities inspector for ontologies plugin for Protégé-4.,” in *International Conference on Biomedical Ontology - ICBO*, 2012.

[58] R. S. Gonçalves, B. Parsia, and U. Sattler, “Performance heterogeneity and approximate reasoning in description logic ontologies,” in *The Semantic Web–ISWC 2012: 11th International Semantic Web Conference, Proceedings*, pp. 82–98, Springer, 2012.

[59] A. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, and C. Wroe, “OWL Pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns,” in *Engineering Knowledge in the Age of the Semantic Web*, pp. 63–81, Springer, 2004.

[60] F. W. Hartel, G. Fragoso, K. L. Ong, and R. Dionne, “Enhancing Quality of Retrieval Through Concept Edit History,” in *AMIA Annual Symposium Proccedings*, pp. 279–283, 2003.

[61] N. Thomas, *NCI Thesaurus - Apelon TDE Editing Procedures and Style Guide*. National Cancer Institute, 2007.

[62] S. Schulz, D. Schober, I. Tudose, and H. Stenzhorn, “The Pitfalls of Thesaurus Ontologization – the Case of the NCI Thesaurus,” in *Proc. of the 2010 AMIA Symposium*, 2010.

[63] W. Ceusters, B. Smith, and L. Goldberg, “A Terminological And Ontological Analysis Of The NCI Thesaurus,” *Methods of Information in Medicine*, vol. 44, no. 4, pp. 498–507, 2005.

[64] G. Fragoso, S. de Coronado, M. Haber, F. Hartel, and L. Wright, “Overview and Utilization of the NCI Thesaurus,” *Comparative and Functional Genomics*, vol. 5, no. 8, pp. 648–654, 2004.

[65] S. de Coronado, L. W. Wright, G. Fragoso, M. W. Haber, E. A. Hahn-Dantona, F. W. Hartel, S. L. Quan, T. Safran, N. Thomas, and L. Whiteman, “The NCI Thesaurus Quality Assurance Life Cycle,” *Journal of Biomedical Informatics*, vol. 42, June 2009.

[66] “The NCI Thesaurus.” http://ncit.nci.nih.gov, June 2015.

[67] R. S. Gonçalves, B. Parsia, and U. Sattler, “Analysing multiple versions of an ontology: A study of the NCI Thesaurus,” in *24th International Workshop on Description Logics*, p. 147, 2011.

[68] M. Horridge and S. Bechhofer, “The OWL API: A Java API for OWL ontologies,” *Semantic Web*, vol. 2, no. 1, pp. 11–21, 2011.

Appendix A

NCIt Files Mapping to NCIt Corpus

In this thesis we used the NCIt Ontology as the case study for evaluating domain modelling bugs. The NCIt Case Study includes in its corpus 112 versions of OWL files from version file name 02.00, published on September 2003, to version file name 13.05d, published on May 2013. The data is publicly available from the NCIt 's website[1], which contains the latest releases of the NCIt Ontology.

The following tables show the mapping between the NCIt 's public file names and this thesis ' ontology identifier names. The ontology identifier names are recorded in the Ontology table part of the MySQL Database set up for the experiments executed in this thesis (See Chapter 3). The results of the experiments conducted in this thesis will be made available as part of The University of Manchester 's OWL Research website[2].

[1]`ftp://ftp1.nci.nih.gov/pub/cacore/EVS/NCI_Thesaurus/archive/`
[2]`http://owl.cs.manchester.ac.uk/research/ncit/`

NCIt Corpus Ont_ID	NCIt Version File Name	Year	Month
1	2	2003	Sep
2	03.10J	2003	Oct
3	03.12A	2003	Dec
4	03.12E	2003	Dec
5	04.02H	2004	Feb
6	04.03N	2004	Mar
7	04.04J	2004	April
8	04.05F	2004	May
9	04.06i	2004	June
10	04.08B	2004	Aug
11	04.09A	2004	Sep
12	04.11A	2004	Nov
13	04.11C	2004	Nov
14	04.12G	2004	Dec
15	05.01D	2005	Jan
16	05.03D	2005	Mar
17	05.05d	2005	May
18	05.06f	2005	June
19	05.07d	2005	July
20	05.09e	2005	Sep
21	05.09g	2005	Sep
22	05.10e	2005	Oct
23	05.11f	2005	Nov
24	05.12f	2005	Dec
25	06.01c	2006	Jan
26	06.02d	2006	Feb
27	06.03d	2006	Mar
28	06.04d	2006	April
29	06.05d	2006	May
30	06.06e	2006	June
31	06.07d	2006	July
32	06.08d	2006	Aug
33	06.09d	2006	Sep
34	06.10d	2006	Oct
35	06.11d	2006	Nov
36	06.12d	2006	Dec
37	07.01d	2007	Jan
38	07.02c	2007	Feb
39	07.03d	2007	Mar
40	07.04e	2007	April
41	07.05e	2007	May
42	07.06d	2007	June
43	07.07c	2007	July
44	07.08d	2007	Aug
45	07.09d	2007	Sep
46	07.10d	2007	Oct
47	07.12a	2007	Dec
48	07.12e	2007	Dec

NCIt Corpus Ont_ID	NCIt Version File Name	Year	Month
49	08.01d	2008	Jan
50	08.02d	2008	Feb
51	08.03d	2008	Mar
52	08.04d	2008	April
53	08.05d	2008	May
54	08.06d	2008	June
55	08.07d	2008	July
56	08.08d	2008	Aug
57	08.09d	2008	Sep
58	08.10e	2008	Oct
59	08.11d	2008	Nov
60	08.12d	2008	Dec
61	09.01d	2009	Jan
62	09.02d	2009	Feb
63	09.03d	2009	Mar
64	09.04d	2009	April
65	09.05d	2009	May
66	09.06e	2009	June
67	09.07e	2009	July
68	09.08e	2009	Aug
69	09.09c	2009	Sep
70	09.10d	2009	Oct
71	09.12d	2009	Dec
72	10.01d	2010	Jan
73	10.02d	2010	Feb
74	10.03h	2010	Mar
75	10.04f	2010	April
76	10.05d	2010	May
77	10.06e	2010	June
78	10.07d	2010	July
79	10.08e	2010	Aug
80	10.10a	2010	Oct
81	10.10d	2010	Oct
82	10.11e	2010	Nov
83	10.12c	2010	Dec
84	11.01e	2011	Jan
85	11.02d	2011	Feb
86	11.03d	2011	Mar
87	11.04d	2011	April
88	11.05e	2011	May
89	11.06d	2011	June
90	11.07d	2011	July
91	11.08e	2011	Aug
92	11.09d	2011	Sep
93	11.10e	2011	Oct
94	11.11d	2011	Nov
95	11.12e	2011	Dec
96	12.01f	2012	Jan
97	12.02d	2012	Feb
98	12.03d	2012	Mar
99	12.04e	2012	April
100	12.05d	2012	May
101	12.06d	2012	June
102	12.07d	2012	July
103	12.08d	2012	Aug
104	12.09d	2012	Sep
105	12.10e	2012	Oct
106	12.11d	2012	Nov
107	12.12d	2012	Dec
108	13.01c	2013	Jan
109	13.02d	2013	Feb
110	13.03d	2013	Mar
111	13.04e	2013	April
112	13.05d	2013	May