A Framework for Change Impact Analysis of Ontology-driven Content-based Systems

Yalemisew M. Abgaz[1], Muhammad Javed[2], Claus Pahl[3]

Centre for Next Generation Localization (CNGL),
School of Computing, Dublin City University, Dublin 9, Ireland
{yabgaz[1]|mjaved[2]|cpahl[3]}@computing.dcu.ie

Abstract. The trend in content-based systems (CBSs) is shifting towards the use of ontologies to semantically enrich the content and increase its accessibility. The growing need of semantically rich content becomes a driving force for building ontology-driven content-based systems (ODCBSs). The building blocks of ODCBSs are ontologies, content and annotations, forming a layered information model. In most ODCBSs, changes in the content, in the ontology or in the annotation are inevitable and are observed on a daily basis. Any change on one layer of the architecture has an impact within the layer and on the other layers. Impact analysis in large multi-ontology CBS is a manual, time consuming and labour intensive process. It is done only when it is necessary. Based on observation and empirical analysis, we propose a conceptual framework for dependency-based impact analysis and identify the possible impacts and their causes, the dependency among entities, their severity and factors affecting impact analysis process in ODCBSs.
Keywords: Ontology evolution, Change impact analysis, Content-based systems, Ontology-driven content-based systems.

1 Introduction

Ontologies became ubiquitous and standard means of embedding semantic information in most of the existing content-based applications [1]. In such applications, ontologies are used to semantically enrich content and services. Many applications are integrated with ontologies using semantic annotation to identify information, process them and reason about subjects of interest. Content-based systems (CBSs) become dependent on ontologies to provide a better service for developers, designers and end-users of such systems. This is achieved by using ontologies to annotate the target content so that both human and computer systems can understand what meaning is exactly conveyed in it [2]. This process leads to the emergence of ontology-driven content-based systems (ODCBSs).

Despite the promising benefits, ODCBSs face challenges. One of the major challenges is the changing nature of content and thus the dynamic evolution of the ontologies that support the ODCBS [3][4]. The interdependence between the content and the ontologies further aggravates the challenge in that, a change in one layer affects entities in the given layer and in all dependent layers. For

relatively large ODCBSs, determining the impacts of a single change operation is difficult, time consuming and often doesn't guarantee a complete solution. To solve this problem, we propose a conceptual framework for dependency-based analysis of impacts of changes in ODCBSs to identify impacts, affected entities and to determine the severity of the impacts. The framework is used to provide terminological and formal guidance for analytical and operational change support. In this context the determination of change impact is a crucial first activity. We used graphs for the formalization of the ODCBS layers to facilitate the dependency analysis and impact determination process.

The term *impact* refers to the effect of change of entities due to the application of a change operation on one or more of the entities in the ODCBS [5][3][4][6]. By *impact analysis* we mean the process of identifying and determining the impacts of a requested change operation on the ODCBSs layers.

Impact analysis identifies the impacts of a change operation before it is permanently implemented. Due to frequent changes in the content and continuing evolution of the ontologies, impact analysis becomes an important step in the evolution of ODCBSs. The core contribution of this paper is a conceptual framework for dependency-based impact analysis using empirical identification of:

- the possible impacts and their categorization.
- the causes of impacts in the content, ontology and annotation layers.
- the dependencies and the types of dependencies that exist between a changing entity and other entities.
- the severity of each of the impacts on the ODCBS and dependent systems.

For a given change request, the knowledge of the above discovered inputs ensures earlier visibility of impacts and smooth evolution by automatically identifying the affected entities and impacts. It guarantees accurate execution of nothing but the desired changes with minimum impacts and it reduces risk on dependent systems by taking prior preventive measures to reduce the impacts.

This paper is organized as follows: Section 2 describes the layers in ODCBSs and Section 3 focuses on graph-based representation of each layer of the ODCBS. Section 4 presents dependencies in ODCBS and section 5 focuses on impacts of changes. Discussion and related work are given in section 6 and conclusion and future work in section 7.

2 Ontology-Driven Content-Based Systems

ODCBSs are systems that use ontologies to semantically enrich the content they provide. The aim of ODCBSs is to facilitate accessibility of content for both humans and machines by integrating semantics in the content using ontologies.

2.1 Layered Architecture of ODCBSs

The ODCBSs is composed of three different layers. The first layer is the ontology layer (represented using OWL), the second is the annotation layer (represented

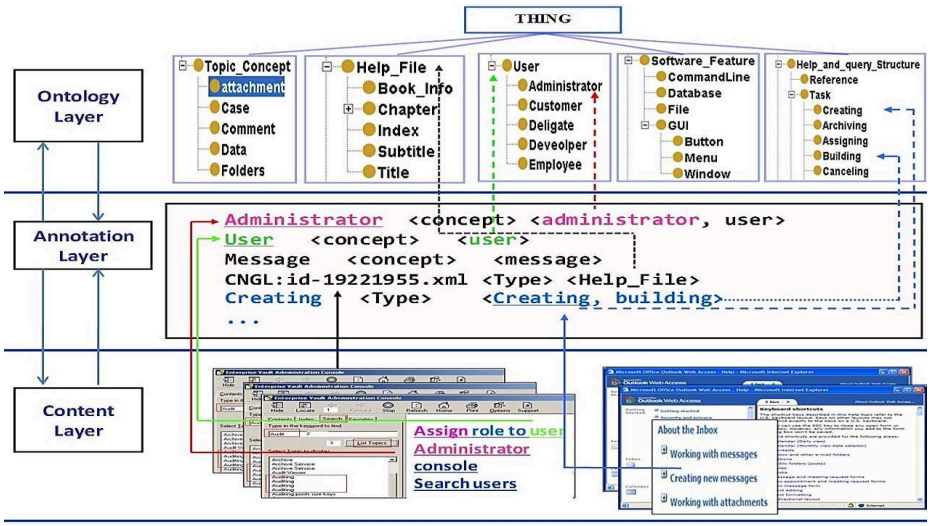

Fig. 1. Layered architecture of ODCBSs

using RDF triples) and the third one is the content layer (set of documents). The layered architecture is presented in (Fig. 1)

Ontology Layer. Ontology is a specification of a shared conceptualization of a domain [7]. This means ontologies provide a common ground for understanding, conceptualization, representation and interpretation of domain concepts uniformly across different systems, languages and formats. They provide a representation of knowledge that allows machines to reason about known facts and generate new knowledge from them.

In our ODCBSs architecture, ontologies become crucial component as many CBSs are integrating ontologies for semantic annotation. A growing number of applications use ontologies to the extent that makes ontologies unavoidable integral parts of the applications.

The ontology layer is subject to change due to a change in specification, representation or conceptualization of knowledge [8]. New concepts are added, existing ones deleted or modified. In frequently evolving domains these changes are numerous and have impact on dependent entities in the ODCBSs.

Content Layer. Content, in this paper, refers to any digital information that is in a textual format that contains structured or semi structured documents, web pages, executable content, software help files etc [9][10]. ODCBSs essentially deal with content in a form of books, web pages, blogs, news papers, software products, documentations, help files reports, publications etc [9].

Content in ODCBSs is a collection of content documents that change frequently. This means new content documents are produced, existing ones are modified, edited or deleted frequently to provide up-to-date information. Such activities have impacts on dependent entities in the overall ODCBS.

Annotation Layer. Annotation is a process of linking content with ontology entities to provide better semantics to the content. The aim of semantic annotation is to explicitly identify concepts and relationships between concepts in the content [1]. In any application that makes use of ontologies, the target content which needs to be semantically enriched is required to have an explicit link, at least to one or more elements in the ontology.

In our ODCBS, the annotation is treated as a separate layer to allow independence of the annotation data from the content, to achieve visibility and better impact analysis. The annotation layer is one of the most interactive and frequently changing layers. There are a number of triples added, modified or deleted in this layer. This layer is highly dependent on both the content and the ontology layer. Any change in the other two layers affect the annotation layer which carries all the semantics related to the content.

In the annotation layer, a document or part of a document is treated as instances of one or more concepts. For example $< CNGL : id{-}19221955.xml, rdf : type, CNGL : Help\_File >$ indicates that "$CNGL : id - 19221955.xml$" is an instance of the concept "$CNGL : Help\_File$" (Fig. 2).

2.2 Running Example

We conducted empirical analysis on database systems, univeristy administration [8] and software help management systems domains [10]. Software help management systems domain is selected to serve as a running example (Fig. 2). Suppose we want to find all the impacts of **Delete Class(*Activity*)** operation. The requested operation is deletion and the target entity is concept *Activity*. To identify the dependent entities, we need to know if the change is applied in a cascaded strategy or not [3][11]. If we choose cascade strategy, meaning if the deletion of the concept *Activity*, deletes all its subclasses, we should identify all the subclasses of *Activity* and their subclasses iteratively, which are { *Archiving, ArchivingEmail, Deleting, DeletingDirectory...* } and save them in a list of dependent entities. We further identify all the axioms {(*Archiving*, subclassOf, *Activity*), (*Deleting*, subclassOf, *Activity*)...} , instances {*CNGL:id-19221955.xml...*} and so on. We identify what kinds of changes are required to each of these dependent entities to make the original change request effective. In the case of cascade delete strategy, we have a set of cascaded change operations like {Delete Concept (*Adding*)..., Delete Instance (*CNGL:id-19221955.xml*)... Delete Axiom (*Archiving*, subclassOf, *Activity*) ... Delete Class (*Activity*)}. The set of change operations on the entities imply their effects. The impacts of these changes, for example, are the removal of the target entities (section 5.1). Once we get the impact set, we attach a severity value to each impact (section 5.2).

In general, the impact varies following the type and the taxonomic position of the target entity, the type of operation and the change strategy implemented. For example, the deletion of the concept *Activity* caused many cascaded operations, due to its structure and the change strategy. However, if the concept *Activity* doesn't have dependent entities or if the change strategy is different, the final change operations will be different and so is the impact set.

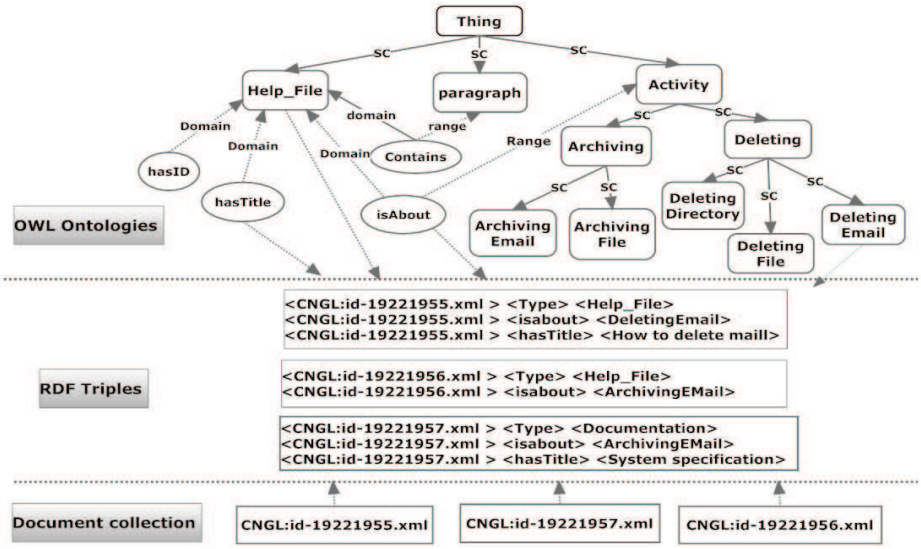

Fig. 2. An example of ODCBS for software help systems

3 Graph-based Representation of ODCBS

The ODCBS can be represented using graph-based formalism. Graphs are selected for their known efficiency and similarity to ontology taxonomy. In our ODCBS, the ontology and the annotation are represented as graphs and the content is represented as a set of documents. The document set serves as a node (of type instances) in the annotation layer.

An ODCBS is represented as graph $G = \{G_o\} \cup \{G_a\} \cup \{Cont\}$ where: G_o is the ontology graph, G_a is the annotation graph and $Cont$ is the content set.

An ontology O is represented by a direct labelled graph $G_o = (N_o, E_o)$ where: $N_o = \{n_{o1}, n_{o2}, \ldots, n_{om}\}$ is a finite set of labelled nodes that represent classes, data properties, object properties etc. $E_o = \{e_{o1}, e_{o2} \ldots, e_{om}\}$ is a finite set of labelled edges and $e_{oi} = (n_1, \alpha, n_2)$ where: n_1 and n_2 are members of N_o and the label of an edge represented by α = {subclassOf, intersectionOf, minCardinality, maxCardinality...}. The labels may indicate the relationship (dependency) between the nodes.

A content represented by $Cont$ can be viewed as a set of documents $D = \{d_1, d_2, d_3....d_n\}$ where: d_i represents a single document or part of a document which can be mapped to nodes in the annotation graph.

An annotation $Anot$ is represented by a direct labelled graph $G_a = (N_a, E_a)$ where: N_a and E_a are finite set of labelled nodes and edges respectively. An edge $E_a = (n_{a1}, \alpha_a, n_{a2})$ where $n_{a1} \in \{Cont\}$ as a subject, $n_{a2} \in \{Cont\} \cup \{O\}$ as an object and $\alpha_a \in \{O\}$ as a predicate. The nodes are mapped to a non-empty string.

The type of any node is given by a function $type(n)$ that maps the node to its type (class, instance, data property, object property...). The label of any edge $e = (n_1, \alpha, n_2)$, which is α , is a string given by a function $label(e)$. All the edges of a node n are given by a function $edges(n)$. It returns all the edges as (n, α, m) where n is the target node and m is any node linked to n via α.

4 Dependency in ODCBSs

Dependency analysis is a process of identifying the artefacts that are dependent on a given entity in an ontology, content or annotation. Dependency analysis identifies all entities that depend on a target entity. Identifying these dependencies and their types has significant contribution to impact analysis process.

4.1 Types of Dependencies

Using the empirical study, we identified the following dependency types that play a major role in the impact analysis process in ODCBSs. We also observed that there is no sharp demarcation between the identified dependency types, thus, they are not mutually exclusive.

Structural Dependency/Semantic Dependency. Structural dependency refers to the syntactic dependency between two nodes. When a node changes, it will have a structural impact on adjacent nodes. Semantic dependency refers to the semantic relation that exists between two nodes. A change in one node e.g. *Activity*, causes a change in the semantic meaning or the interpretation of the dependent nodes *(Archiving and Deleting)*.

Direct Dependency/Indirect Dependency. Direct dependency is the dependency that exist between two adjacent nodes(n_1, n_2). This means, there is an edge $e_i = (n_1, \alpha, n_2)$. Indirect dependency is a dependency of a node on another by a transitive or intermediate relationship. There exist a set of intermediate edges $(n_1, \alpha, n_x)(n_x, \alpha, n_y)...(n_z, \alpha, n_2)$ that link the two nodes. For example, in Fig. 2 there is a direct dependency between *Activity* and *Deleting* and indirect dependency between *Activity* and *Deleting_File*

Total Dependency/Partial Dependency. Total dependency refers a dependency when a target node depends only on a single node (articulation node). Partial dependency refers to a dependency when the existence of a node depends on more than one node.

4.2 Dependency within and among Layers

Dependency in the Ontology Layer. A change of an entity in one ontology first affects the dependent entities within the ontology. Identifying the dependencies in this layer is a crucial step. These dependencies are identified based on the inheritance (such as *is-a* relationships) association (such as *has* relationships) and so on. There is also dependency across ontologies. We present one of the empirically identified dependencies from our case study.

Concept-Concept Dependency: Given two class nodes c_i and c_j in G_o, c_i is dependent on c_j represented by $dep(c_i, c_j)$, if there exist an edge $e_i = (n_1, \alpha, n_2) \in G_o$ such that $(n_1 = c_i) \wedge (n_2 = c_j) \wedge (label(e_i) = \text{“}SubclassOf\text{”}) \wedge (type(n_1) = type(n_2) = \text{“}Class\text{”})$. Concept-concept dependency is transitive.

Dependency in the Annotation Layer. In this layer we have two directions of dependency. The first refers to the dependency of the annotation on the content layer (Content-Annotation Dependency). An annotation a_i in the annotation layer is dependent on d_i in the content layer, represented by $dep(a_i, d_i)$, if there exist an edge $e_a = \{n_{ai}, \alpha_a, n_{aj}\} \in G_a$ such that $(n_{ai} = d_i) \vee (n_{aj} = d_i)$. This means a_i is dependent on document d_i if the document is used as a subject or an object of the annotation triple.

The second refers to the dependency of the annotation on the ontology (Ontology-Annotation Dependency). An annotation a_i in the annotation layer is dependent on o_i in the ontology layer, represented by $dep(a_i, o_i)$, if there exist an edge $e_a = \{n_{ai}, \alpha_a, n_{aj}\} \in G_a$ such that $(\alpha_a = o_i) \vee (n_{aj} = o_i)$.

Dependency in the Content Layer. Intra content dependency (Content-Subcontent Dependency) is a dependency that exists between a document and its subsections. This includes the dependency of section, title, paragraph, step, procedure etc on a containing document. Whenever the content in the documents are updated, for example, deletion of a section, addition of steps etc, affect all the section and documents related to the document.

5 Impact of Changes in ODCBSs

Changes in ODCBS have diverse impacts on the individual entities of the layers and on the overall ODCBS. Impact analysis identifies the possible impacts of proposed changes and determine the severity of the impacts [12][6][13]. Determining the impacts that exist in the ODCBS and deciding the severity of the impacts are essential steps for impacts analysis.

5.1 Types of Ontology Change Impact

In ODCBS we can categorize impacts using different criteria. The categorization paves a way to better understand impacts in ODCBS and makes the analysis and the determination process understandable and suitable for implementation.

Structural and Semantic Impact. Structural impact is an impact that changes the structural relationship between the elements of the ODCBS. Structural changes are the main reasons for structural impacts. Structural changes include any atomic or composite changes[3] that are applied on concepts, properties, axioms and restrictions. Structural impact occurs when we request a change that affects the taxonomy of the existing ODCBS. Semantic impact occurs due to a change in the interpretation of entities due to structural changes. Structural and semantic changes are discussed in [5] and the impacts are discussed in [10].

Addition and Deletion Impact. The categorization of impacts of addition and deletion became visible in the empirical study. In ODCBSs, the impacts of

addition operation are different from the impacts of deletion operation. Furthermore, the complexity of the impacts differ one another. In such situation it is intuitive to treat the operations and their impacts separately.

Ontology, Annotation and Content Impact. Impacts can further be divided based on the target layer. This categorization allows us to know which layers are affected by the change operation. Impacts in the ontology layer include all impacts on the entities defined in the given ontology. Such impacts need careful treatment as they further affect the annotation layer. Impacts in the annotation layer primarily revolve around the triples. However, a triple contains the subject (usually a reference to the content), the predicate (usually a property in the otology) and the object (usually a reference to the content or the ontology). Thus, impact analysis in the annotation layer makes use of these three elements and tries to find what impacts the change operation will have on them. Impacts in the content layer concentrate around the documents. The addition, deletion or the modification of the content or part of the content is treated as an impact and affects the other two layers.

$\mathcal{A}$Box and $\mathcal{T}$Box Impacts. Impacts of a change operation can be viewed from the perspective of the kind of statement it affects. Change operations may have an impact on the $\mathcal{A}$Box or $\mathcal{T}$Box statements. $\mathcal{T}$Box statements are affected by operations that change the concepts and axioms related to the terminology in the ontology. The impact of such change operation concentrates around the satisfiability of the terminologies in the $\mathcal{T}$Box and identifying them helps us to pinpoint the causes of contradiction. $\mathcal{A}$Box statements are affected by operations that change the axioms related to annotation instances (individuals) in the assertion box. The impact of operations on the $\mathcal{A}$Box axioms may result invalidity (unable to interpret a give instance with respect to a given ontology) [5].

Impact analysis is mainly affected by the change strategy implemented at the time of evolution. The content engineer may choose to delete all orphaned entities or link them to their parents or to the root class. The different types of dependencies are crucial at this stage. For example, direct dependency is used when attach to root strategy is used. Total dependency is used when cascade delete strategy is used. Partially dependent entities remain intact in the system.

5.2 Severity of Change Impacts

Severity is defined as the extent of impact of a change operation in the ODCBS. The impact is measured qualitatively using consistency and validity, or quantitatively using number of change operations required, ontology elements affected and cascaded effect on dependent entities.

The impact can be on the structure or on the semantic, satisfiability or validity of the existing ontology. We analyzed how different change operations impact the ODCBS. This gives us a better understanding of which operations under what condition have a more severe impact. We used Operation Severity Function (OSF) that maps operations to severity values on a scale of 0 to 100 based on user defined configuration. To indicate the severity as a qualitative scale, we categorized them into four scales: less impact (0-25), medium impact

(26-50), high impact (51-75) and crucial impact (76-100) [10]. Based on this scale the user can determine the severity of an impact relative to his ODCBS.

6 Discussion and Related Work

Using a software help management system [10], we tested our approach empirically. From the case study, we found out that the proposed solution is useful and feasible to analyze impacts of changes in ODCBSs. To our knowledge, there are few research conducted to analyze the impacts of changes in ODCBSs. But there are significant research in the area of software change impact analysis in general [14] [6] [15].

The author in [14] conducted change impact analysis on commercial-Off-The-Shelf software. They identified different reasons for software change and classify software impacts as direct or indirect, and structural or semantic impacts. They further conducted structural analysis and semantic analysis using reachability graphs by implementing transitive closure algorithms. They focus on the syntactic relationship between software modules where as we focus on structural and semantic changes with detailed semantics. In [6] the authors presented a knowledge-based system for change impact analysis on software architecture. They proposed an architectural software component model on which they defined change propagation process and used graphs to capture architecture elements and their relationships. The authors conducted impact analysis using rules that define change propagation. Their work is similar to ours but with a significant difference in the domain and in the impact determination approach.

The work in [5] discusses consistent evolution of OWL ontologies with the aim of guaranteeing consistency. Their work identifies structural, logical and user-defined consistency, but focuses on determination of validity of instances, whereas our work focuses on the overall impact analysis of change operations.

7 Conclusion and Future Work

Based on our empirical analysis, we identified different dependencies that exist within and among ODCBS layers. We identified dependencies and impacts in a conceptual framework then further categorized them based on different criteria that will serve as an input for the impact analysis process. We also investigated the severity of the impacts.

The contribution of our work is the empirical observation of dependencies in ODCBS, types of impacts and factors affecting impact analysis. This will enable us to ensure earlier visibility of impacts of changes prior to their implementation, automatic capturing and presentation, accurate determination and reducing their impacts on dependent systems. Our next step will further extend the work to optimize the implementation of change operations to ensure minimum and less severe impact.

Acknowledgment. This material is based upon works supported by the Science Foundation Ireland under Grant No. 07/CE/I1142 as part of the Centre for Next Generation Localisation (www.cngl.ie) at Dublin City University (DCU).

References

1. Uren, V., Cimiano, P., Iria, J., Handschuh, S., Vargas-Vera, M., Motta, E., Ciravegna, F.: Semantic annotation for knowledge management:requirements and survey of the state of the art. Web Semantics: Science, Services and Agents on World Wide Web. **4**(1) (2006) 14–28
2. Reeve, L., Han, H.: Survey of semantic annotation platforms. In: SAC '05: Proceedings of the 2005 ACM symposium on Applied computing. (2005) 1634–1638
3. Stojanovic, L.: Methods and tools for ontology evolution. PhD thesis, University of Karlsruhe (2004)
4. Plessers, P., De Troyer, O., Casteleyn, S.: Understanding ontology evolution: A change detection approach. Web Semantics: Science, Services and Agents on the World Wide Web. **5**(1) (2007) 39–49
5. Qin, L., Atluri, V.: Evaluating the validity of data instances against ontology evolution over the semantic web. Information and Software Technology. **51**(1) (2009) 83–97
6. Hassan, M.O., Deruelle, L., Basson, H.: A knowledge-based system for change impact analysis on software architecture. In: Research Challenges in Information Science (RCIS), 2010 Fourth International Conference on. (may 2010) 545 –556
7. Gruber, T.R.: A translation approach to portable ontology specifications. Knowl. Acquis. **5**(2) (1993) 199–220
8. Javed, M., Abgaz, Y., Pahl, C.: A pattern-based framework of change operators for ontology evolution. In: On the Move to Meaningful Internet Systems: OTM 2009 Workshops. Volume 5872 of Lecture Notes in Computer Science. (2009) 544–553
9. Gruhn, V., Pahl, C., Wever, M.: Data model evolution as basis of business process management. In: Proceedings of the 14th International Conference on Object-Oriented and Entity-Relationship Modelling. OOER '95, London, UK, Springer-Verlag (1995) 270–281
10. Abgaz, Y., Javed, M., Pahl, C.: Empirical analysis of impacts of instance-driven changes in ontologies. In: On the Move to Meaningful Internet Systems: OTM 2010 Workshops. Lecture Notes in Computer Science. (2010)
11. Haase, P., Stojanovic, L.: Consistent evolution of owl ontologies. In: Proceedings of the Second European Semantic Web Conference, Heraklion, Greece. (2005)
12. Li, L., Offutt, A.J.: Algorithmic analysis of the impacts of changes to object-oriented software. In: Proceedings of the International Conference on Software Maintenance, IEEE (1996) 171–184
13. Arnold, R.S., Bohner, S.A.: Impact analysis - towards a framework for comparison. In: Proceedings of the Conference on Software Maintenance. ICSM '93, Washington, DC, USA, IEEE Computer Society (1993) 292–301
14. Bohner, S.: Extending software change impact analysis into cots components. In: Software Engineering Workshop, 2002. Proceedings. 27th Annual NASA Goddard/IEEE. (dec. 2002) 175 – 182
15. Sherriff, M., Williams, L.: Empirical software change impact analysis using singular value decomposition. In: Proceedings of the 2008 International Conference on Software Testing, Verification, and Validation, Washington, DC, USA, IEEE Computer Society (2008) 268–277