Review of Ontology Evolution Process

Helna Wardhana
Computer Science & Electronic Department, Gadjah Mada University, Yogyakarta STMIK Bumigora Mataram, Lombok, Indonesia

Ahmad Ashari
Computer Science & Electronic Department, Gadjah Mada University, Yogyakarta

Anny Kartika Sari
Computer Science & Electronic Department, Gadjah Mada University, Yogyakarta

ABSTRACT
Ontologies are formal artifacts that are designed to represent the knowledge related to a specific or generic domain in terms of the relevant concepts, relationships between these concepts and the instances of these concepts. Ontology evolution can be defined as the process to adapt and change the ontology in a timely and consistent manner. In this paper, we present a brief description of ontology evolution process of recent research.

General Terms
Ontology change, knowledge representation

Keywords
Ontology, ontology evolution, semantics of change

1. INTRODUCTION
Information semantics and semantic interoperability between applications, systems, and services are mostly based on ontology [1]. It is increase usage in information systems and knowledge sharing systems raises the importance of ontology maintenance. Ontologies are formal artifacts that are designed to represent the knowledge related to a specific or generic domain in terms of the relevant concepts, relationships between these concepts and the instances of these concepts. With rising importance of knowledge interchange, many industrial and academic applications have adopted ontologies as their conceptual backbone. Ontologies, to be effective, need to change as fast as the parts of the world they describe. A modification in one part of the ontology may generate many inconsistencies in other parts of the same ontology, in the ontology-based instances as well as in depending ontologies and applications that are based on this ontology [2]. Ontology evolution can be defined as the process to adapt and change the ontology in a timely and consistent manner.

2. PRELIMINARIES
2.1 Ontology
An ontology is an explicit specification of a conceptualization [3]. According to [4] an ontology is a representation vocabulary, often specialized to some domain or subject matter. More precisely, it is not the vocabulary as such that qualifies as an ontology, but the conceptualizations that the terms in the vocabulary are intended to capture. An ontology became important because without ontologies, or the conceptualizations that underlie knowledge, there cannot be a vocabulary for representing knowledge [4]. Thus, the first step in devising an effective knowledge representation system, and vocabulary, is to perform an effective ontological analysis of the field, or domain. Weak analyses lead to incoherent knowledge bases. According to [1] in computer science, ontology is defined as formal and explicit specifications of a shared conceptualization of a domain of discourse and is the main driving force behind Semantic Web vision.
Ontologies enable knowledge to be made explicit, formalize the relevant underlying view of the world (domain model) and make such models machine processable and interpretable [5]. In [6], ontologies to be effective, need to change as fast as the parts of the world they describe. There are two main challenges in adapting ontologies. The evolution of ontologies should reflect both the changing interests of people and the changing data, for example the documents stored in a digital library.

2.2 Ontology Change
Ontology change refers to the generic process of changing an ontology in response to a certain need. Some of the aspects that will initiate a change when requested for accommodation in the ontology [1]:

- *New Concept:* This is the most common change in any ontology. New concepts emerge and have to be accommodated in the concept hierarchy.
- *Concept with Changed Properties:* This is the case when the concept in focus is already present in the ontology but its properties and restrictions are different from those associated with existing concepts.
- *Simple vs. Aggregated Concept:* The concept in focus might be a combination of two or more existing concepts (or vice versa). The ontology framework shall preferably detect and act accordingly to accommodate the change.
- *Concept vs. Property:* Different modeling approaches are followed by ontology engineers for building ontologies. One such case is modeling the same concept either as a class in OWL or as a property of some other existing class. For example, the concept *Luxury_Vehicle* could be a separate subclass of *Vehicle* or could be modeled as property of the concept *Vehicle*.
- *Concept with Changed Hierarchy:* Different modeling approaches may fix the same concept in different hierarchical locations in two different ontologies.

Following figure 1 shown a classification of ontology change subfields.

Any field handling any type of change or related issues

(Ontology Change)

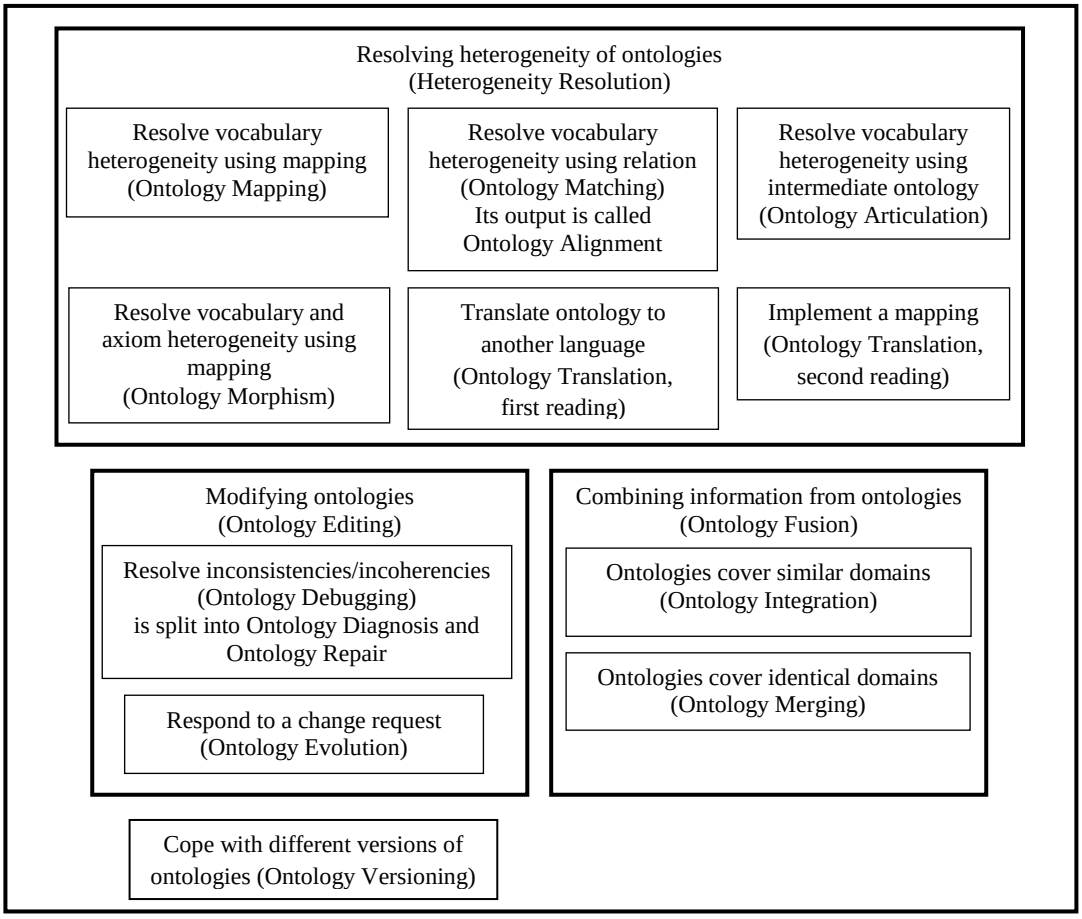

Figure 1. The Rotation of Ontology Evolution

2.3 Ontology Evolution

Evolution is an intrinsic part of the Semantic Web. Alterations in a particular domain, changes to user requirements or corrections of design flaws, they all may induce changes to the corresponding ontologies. Moreover, changes to one ontology may have implications on many depending artifacts [7]. [8] see ontology evolution as the process to 'adapt and change the ontology in a timely and consistent manner'. Ontology evolution defined as a process aiming to 'respond to a change in the domain or its conceptualization' by implementing a set of change operators over a source ontology [9]. The recently compiled NeOn Glossary of ontology engineering tasks states that ontology evolution is 'the activity of facilitating the modification of an ontology by preserving its consistency [9]. According to [2] ontology evolution is the timely adaptation of an ontology to the arisen changes and the consistent propagation of these changes to dependent artefacts. The complexity of ontology evolution increases as ontologies grow in size, so a structured ontology evolution process is required. Ontology evolution is defined as the formal interpretation of all change requirements captured from different sources, the application of changes to the ontology, and their propagation to dependent artifacts while preserving consistency. Dependent artifacts include objects referenced by the ontology and, dependent ontologies and applications. In [10], ontology evolution means modifying or upgrading the ontology when there is a certain need for change or there comes a change in the domain knowledge. Ontology evolution is the timely adaptation of an ontology to changed business requirements, to trends in ontology instances and patterns of usage of the ontology based application, as well as the consistent management/propagation of these changes to dependent elements [11]. Ontology evolution is a complex problem: Besides identifying change requirements from several sources (modeled domain, usage environment, internal conceptualization, etc.), the management of a change –from a request to the final validation and application– needs to formally specify the required change, to analyze and resolve change effects on ontology, to implement the change, and to validate its final application [12].

Why would someone want to develop an ontology? Some of the reasons are: To share common understanding of the structure of information among people or software agents, To enable reuse of domain knowledge, To make domain assumptions explicit, To separate domain knowledge from the operational knowledge, To analyze domain knowledge [13].

3. ONTOLOGY EVOLUTION PROCESS

An ontology is a 'specification of a shared conceptualization of a domain' [3] and therefore needs to change (i.e., to evolve) whenever changes occur in the underlying domain or in its conceptualization. According [1] The current ontology evolution techniques have several hidden weaknesses which

are still needed to be unfolded for the purpose of automatic ontology evolution and minimizing its after effects. One major weakness is that the specification of new changes due to change in domain knowledge, resolving inconsistencies because of new changes (selecting deduced changes from available alternatives), and also undo and redo in case we want to recover the ontology are all done manually [23]. In order to automate the process of ontology evolution, we need to automate all the mentioned tasks. This automation is important because human intervention is time consuming and error prone. In addition to these issues, the process of evolution also brings consequent effects on dependent applications and services using the evolving ontology, which must be minimized [15, 24, 25].

The process starts with capturing changes either from explicit requirements or from the result of change discovery methods. Next, changes are represented formally and explicitly. The semantics of change phase prevents inconsistencies by computing additional changes that guarantee the transition of the ontology into a consistent state. In the change propagation phase, all dependent artifacts (ontology instances on the Web, dependent ontologies, and application programs using the changed ontology) are updated. During the change implementation phase, required and induced changes are applied to the ontology in a transactional manner. In the change validation phase, the user evaluates the results and restarts the cycle if necessary. According to [6], there are six-phase evolution process, the phases being: (1) change capturing, (2) change representation, (3) semantics of change, (4) change propagation, (5) change implementation, and (6) change validation. The following figure 2 show the rotation of ontology evolution.

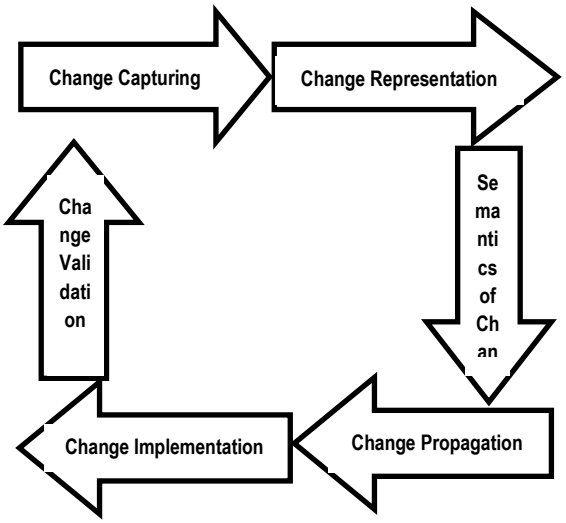

Figure 2. The Rotation of Ontology Evolution

Ontology over time needs to be updated to accommodate changes in domain of knowledge, user requirements, and to incorporate incremental improvement in the system. The following table 1 show different ontology evolution approaches.

Table 1. Varian of Ontology Evolution Process

No	Refs	Stages of Ontology Evolution Process
1.	[14] [6] [12]	Change Capturing, Change Representation, Semantics of Change, Change Propagation, Change Implementation, Change Validation
2.	[2] [11] [10] [1] [15]	Change Capturing, Change Representation, Semantics of Change, Change Implementation, Change Propagation, Change Validation
3.	[7]	Evolution on Request (Change Request, Consistency Maintenance, Change Detection, Change Recovery, Change Implementation) and Evolution in Response (Change Detection, Cost of Evolution, Version Consistency)
4.	[16] [17]	Requesting a change (change representation/ change request), Planning the change (change impact analysis), Implementing the change (change propagation, restructuring, inconsistency management), Verifying and Validating the change (formal verification, testing, debugging, quality assurance)
5.	[9]	Detecting the Need for Evolution, Suggesting Changes, Validating Changes, Assessing Impact, Managing Changes

Furthermore, to facilitate our understanding of ontology evolution, we present the following table 2.

Table 2. Ontology Evolution Filling Table [17]

	Ontology Evolution
Definition	Ontology evolution refers to the activity of facilitating the modification of an ontology by preserving its consistency, it can be seen as consequence of different activities during the development of the ontology
Aim	The aim of ontology evolution is to modify and to change an ontology based on arisen requirements
Input	An ontology in a consistent state
Output	An ontology in a consistent state with the proposed changes applied
Who	All ontology engineers that have to perform changes/updates to a deployed ontology
When	Normally it occurs after the ontology has been deployed and need to be updated.
Why	Because there is a certain requirement for change or there comes a change in the domain knowledge, consequently need to share common understanding of the structure of information among people or software agents
How	By implementing a set of change operators (add, delete or modify) over a source ontology

3.1 Change Capturing

The ontology evolution process begins with change capturing phase, it means to capture all the required changes to be applied to an ontology [1]. Capturing changes could be done either from explicit requirements or from the result of change discovery methods, which cause changes from patterns in data and usage [6]. Explicit requirements are produced by ontology engineers who want to adjust the ontology to novel requirements or by the end-users who present the explicit feedback about the utilization of ontology entities. The changes generating from those requirements are called top down changes whereas implicit requirements leading to so-called bottom-up changes are reflected in the behavior of the

system and can be discovered only through the analysis of this behavior [6].

According to [2], there are two types of change discovery, namely usage-driven and data-driven change discovery. Usage-driven changes result from the usage patterns created over a period of time. Once ontologies reach certain levels of size and complexity, the decision about which parts remain relevant and which are outdated is an immense task for ontology engineers. Usage patterns of ontologies and their metadata allow the detection of often or less often used parts, thus reflecting the interests of users in parts of ontologies. They can be derived by tracking querying and browsing behaviors of users during the application of ontologies. Data-driven change discovery can be explained as the task of deriving ontology changes from modifications to the knowledge representation it has been constructed from [2]. While usage-driven changes arise out of usage patterns of the ontology, data-driven changes are generated by modifications of the reference-data such as text documents or a database which contains the knowledge modeled by the ontology [6]. [11] and [15] first provided the functional requirements for the system to properly interact with the underlying model and also provided multiple types of changes related to class, properties, hierarchy, instances, and restrictions.

The change capturing phase was phase where changes to be applied to the ontology are identified. Three types of changes are identified based on usage-driven change discovery (i.e., derived from user behavior), data-driven discovery (i.e., derived from changes to the ontology instances) and structure-driven change discovery where changes are derived from the analysis on the structure of the ontology [9].

3.2 Change Representation

The second phase in ontology evolution is change representation. Change representation is the phase where all the required changes are represented using formal representational format ([1], [10]) or following a specific model [9]. According to [6] and [2], the set of ontology change operations depends heavily on the underlying ontology model. Most existing work on ontology evolution builds on frame-like or object models, centered around classes, properties, etc. To complete those changes [6], the required changes have to be identified and represented in a suitable format which means that the change representation needs to be defined for a given ontology model. The identified changes are reflected according to the specification of KAON language. Change can be represented on three granularity levels: elementary change, composite change, and complex change ([2], [12], [15]). A taxonomy of elementary changes is derived as the cross product of the set of entities of the ontology model and the meta-change transformations *add* and *remove*. The author also mentions that this level of change representation is not always appropriate and therefore introduces the notion of composite changes. A composite change is an ontology change that modifies (creates, removes or changes) one and only one level of neighborhood of entities in the ontology. Examples for these composite changes would be: “Pull concept up”, “Concept Copy”, “Split Concept”, etc. Furthermore, a complex change is an ontology change that can be decomposed into any mix of at least two elementary and composite ontology changes ([2], [15]).

[2] derives a set of ontology changes for the KAON ontology model. Information about changes can be represented in many different ways [18]. The author describes different representations and propose a framework that integrates them. [19] describes a set of changes for the OWL ontology language, based on an OWL meta-model. The author considers also modify operations in addition to *Delete* and *Add* operations. Another form of change representation for OWL is defined by [8], who follow an ontology model influenced by Description Logics, which treats an ontology as a knowledge base consisting of a set of axioms. Models for change representations for other ontology language exist, too: a formal method for tracking changes in the RDF repository is proposed in [20].

3.3 Semantics of Change

The third phase in ontology evolution process is semantics of change, i.e. the phase that enables the resolution of ontology changes in a systematic manner by ensuring the consistency of the ontology [15]. The goal from this phase is to evaluate and resolve change effects in a systematic manner by ensuring the consistency of the whole ontology [12].

According to [1], semantics of change is the phase where the effects of the required changes are tested on ontology for its consistency and if required then some deduced changes are also included in the change request to avoid conflicts.

The ontology change operations need to be managed such that the ontology remains consistent throughout [6]. The consistency of an ontology is defined in terms of consistency conditions, or invariants that must be satisfied by the ontology. The meaning of consistency depends heavily on the underlying ontology model. It can for example be defined using a set of constraints or it can be given a model-theoretic definition. In the following we provide an overview of various notions of consistency and approaches for the realization of the changes. Ontology consistency is defined as following: “A single ontology is defined to be consistent with the respect to its model if and only if, it preserves the constraints defined for underlying ontology model” [2]. The global approach focuses on KAON ontology. In [8], the authors describe the semantics of change for the consistent evolution of OWL ontologies, considering the structural, logical, and user-defined consistency conditions:

a) *Structural Consistency* ensures that the ontology obeys the constraints of the ontology language with respect to how the constructs of the ontology language are used.

b) *Logical Consistency* regards the formal semantics of the ontology: viewing the ontology as a logical theory, an ontology as logically consistent if it is satisfiable, meaning that it does not contain contradicting information.

c) *User-defined Consistency*: Finally, there may be definitions of consistency that are not captured by the underlying ontology language itself, but rather given by some application or usage context. The conditions are explicitly defined by the user and they must be met in order for the ontology to be considered consistent.

[2] explains and compares two approaches to verify ontology consistency, called a posteriori verification and a priori verification. The first approaches, a posteriori verification, where first the changes are executed, and then the updated ontology is checked to determine whether it satisfies the consistency constraints. Whereas a priori verification, which defines a respective set of preconditions for each change. It must be proven that, for each change, the consistency will be maintained if an ontology is consistent prior to an update and the preconditions are satisfied.

The semantics of change phase, during which syntactic and semantic inconsistencies that could arise as a result of the changes are addressed [9]. Yet another problem is that the change may introduce inconsistency in the ontology. According to [21], an inconsistency is any situation in which a set of descriptions does not obey some relationship that should hold between them. The relationship between descriptions can be expressed as a consistency rule against which the descriptions can be checked.

3.4 Change Propagation

The fourth phase in ontology evolution process is change propagation. According to ([12], [1], [22]) the change of propagation phase aims to propagate ontology changes to the possible dependent artifacts, applications and services in order to preserve the overall consistency. The task of this phase is to bring automatically all dependent artefacts into a consistent state after an ontology update has been performed [2]. These artifacts can be ontologies reused or extended by the evolved ontology or distributed applications. Propagation consists in tracking and broadcasting applied changes. Synchronization approaches proposed are described in detail in ([2], [14]).

According to [6], the task of the change propagation phase of the ontology evolution process is to ensure consistency of dependent artefacts after an ontology update has been performed. These artefacts may include dependent ontologies, instances, as well as application programs using the ontology. Change propagation allowing the update of outdated instances as well as recursively reflecting changes in referenced ontologies in the case of interconnected ontologies [9].

An approach for evolution in the context of dependent and distributed ontologies have been provided by [14]. In [14], definition of *Dependent Ontology Consistency* is state of an ontology if the ontology itself and all its included ontologies, observed alone and independently of the ontologies in which are reused, are single ontology consistent. [14] compared *Push-based* and *Pull based* approaches for the synchronization of dependent ontologies. The authors follow a push-based approach for dependent ontologies on one node and present an algorithm for dependent ontology evolution. Push-based synchronization is a variant of immediate conversion. Furthermore, for the case of multiple ontologies on multiple nodes, the authors define *Replication Ontology Consistency* (An ontology is replication consistent if it is equivalent to its original and all its included ontologies (directly and indirectly) are replication consistent). The authors follow a pull-based approach for the synchronization between originals and replicas. With pull-based synchronization, the changes are propagated at explicit request, which implies a deferred approach.

According to [16], during the change planning phase, the impact of the change has been analyzed, and it may turn out that a local change will propagate to many different types of dependent artefacts. Based on the cost and impact analysis, the ontology engineer might consider to cancel the change or not.

3.5 Change Implementation

The fifth phase in ontology evolution process is the implementation of change, coupled with user interaction for approving or cancelling changes [9]. Change implementation is the phase where the complete change request (modified) is executed on the ontology ([1], [10]).

In ([6], [2], [15]), the role of the change implementation phase of the ontology evolution process is: (i) to inform an ontology engineer about all consequences of a change request, (ii) to apply all the (required and derived) changes, and (iii) to keep track of performed changes. Furthermore, we describe these functionalities in detail.

- *Change Notification*: In order to avoid performing undesired changes, a list of all implications for the ontology and dependent artefacts should be generated and presented to the ontology engineer, who should then be able to accept or abort these changes.
- *Change Application*: The application of a change should have transactional properties, that is (A) Atomicity, (C) Consistency, (I) Isolation, and (D) Durability. The approach of [2] realizes this requirement by the strict separation between the request specification and the change implementation. This allows the set of change operations to be easily treated as one atomic transaction, since all the changes are applied at once.
- *Change Logging*: There are various ways to keep track of the performed changes. [2] provides an evolution log based on an evolution ontology for the KAON ontology model. KAON Change logging is based on two specific notions: *evolution ontology* and *evolution log*. Evolution ontology defines a model of applicable changes on an ontology, facilitating the management of these changes. Evolution log describes applied-change historic through chronological information sequences about each change. It holds knowledge about the ontology development and maintenance. Modeling change (evolution ontology) and their application historic (evolution log) help in synchronizing the ontology evolution. If a c hange needs to be cancelled, evolution log based on a formal change model, helps in guiding revoke operations. The evolution ontology covers the various types of changes, dependencies between changes (causal dependencies as well as ordering), as well as the decision-making process [2].

According to [12], this phase consists in the physical implementation of the required change and the derived changes resolving it. First, changes are notified to the ontology engineer to be approved and then, applied. Besides, all performed changes are logged in order to support recovery facilities.

3.6 Change Validation

The last phase in ontology evolution process is the validation phase, which checks that the performed changes led to a valid (or desirable) result, and allows the user to undo such changes if this is not the case [9]. In [10], this phase called change of verification, sub task validates the subject ontology to confirm that the requested changes have been committed to the ontology. This phase consists in the final validation of the applied changes. It ensures the reversibility of the changes if they are finally disapproved by users (may be due to no convincing impacts, divergent points of view in collaborative context, etc.), the rationale explanation of changes, and their usability ([2], [12]).

According to [6], there are numerous circumstances where it can be desirable to reverse the effects of the ontology evolution, as for example in the following cases:

- The ontology engineer may fail to understand the actual effect of the change and approve a change which should not be performed.

- It may be desired to change the ontology for experimental purposes.
- When working on an ontology collaboratively, different ontology engineers may have different ideas about how the ontology should be changed.

It is the task of the change validation phase to recover from these situations. Change validation enables justification of performed changes or undoing them at user's request. Consequently, the usability of the ontology evolution system is increased.

In [17], before the ontology is considered evolved completely, the last step deals with assessing questions whether the right ontology is built and whether it is built in the right way. During this assessment, usually not only the ontology originally modified is verified in isolation, but in general, this activity can include the verification of other artefacts related to the ontology to ensure that they were not changed in a wrong way or they have an unexpected behavior. [16] have been mentioned that activity in the change process has to do with *verification* and *validation*. Verification addresses the question "did we build the system-right?", whereas validation addresses the question "did we build the right system?". A wide scale of different techniques has been proposed to address these questions, including: testing, formal verification, debugging and quality assurance.

4. DISCUSSION

According to [1], the evolution in ontology is mainly of two types *i.e.*, *Ontology Population* and *Ontology Enrichment*. *Ontology Population* is when we get new instances for concept that is already provided in the ontology. Here only the new instance(s) of the concept is introduced and the ontology is populated. *Ontology Enrichment* is when we get changes in the structure of ontology. For example, when we get new concept(s), which is totally new for the ontology or the concept does have some sort of changes from its counter concept in the ontology.

Based on the explanation in section 3, we can conclude several methods/approaches used in each phase in ontology evolution. The following table 3 shows methods proposed on ontology evolution process.

Table 3. Summary of Methods on Ontology Evolution Process

Ontology Evolution Process	**Used Methods/Approaches**	**Refs**
Change Capturing	Usage-driven change discovery, data-driven change discovery and structure-driven change discovery	[9], [2], [14]
Change Presentation	KAON ontology	[2]
	OWL ontology	[19]
	Ontology of change operation (the kernel of the framework)	[18]
Semantics of Change	the structural, logical, and user-defined consistency conditions for consistent evolution of OWL ontologies	[8]
	A posteriori verification and a priori verification	[2]
Change Propagation	*Push-based* and *Pull based* synchronization	[14]
	the creation and preservation of sub-ontologies to deal with the frequent changes in health ontologies	[23]
Change Implementation	Change Notification, Change Application, Change Logging	[6], [2], [15]
Change Validation	Justification of the changes	[2]
	Relevance of the changes with respect to the ontology	[24]
	testing, formal verification, debugging and quality assurance	[16]

Furthermore, the following is a brief explanation of used method in each phase.

In [9], change capturing phase is equal to detecting the need for evolution. The goal of the Detecting the Need for Evolution stage is to detect whether new concepts and relations should be added to the ontology, or whether some ontology elements can be deleted. [2] defines data-driven ontology evolution as the process of discovering ontology changes based on the analysis of the ontology instances, for example, by relying on data mining techniques. Another type of change detection defined by [2] is structure driven, where the evolution is initiated based on the analysis performed on the ontology structure using a set of heuristics. For example, 'if all sub concepts have the same property, the property may be moved to the parent concept', or 'a concept with a single sub concept should be merged with its sub concept'[2].

[18] have developed an ontology of change operations for the OWL knowledge model as an example of a common language for the interaction of tools and components in their framework. The operations in this ontology are the elements for the specification of a *transformation set*. The ontology consists of two parts. The basis is an ontology of basic change operations and there is an extension that defines complex change operations. [18] chose the set of basic operations in such a way that the required commitment is minimal, while the set is still rich enough to capture enough knowledge about the change to derive new information.

According to [2], application of an elementary change to an ontology can induce inconsistencies in other parts of the ontology. The author distinguishes structural and semantic inconsistency. Structural inconsistency arises when the ontology model constraints are invalidated (e.g. undefined entities at the ontology or instance level are used). Whereas semantic inconsistency arises when the meaning of an ontology entity is changed due to the changes performed in the ontology.

[14] have been defined there are two ways of propagating changes from the changed ontology to all ontologies that include it, called push-based approach and pull-based approach. *Push-based approach*: Changes from the changed ontology are propagated to dependent ontologies as they happen. *Pull based approach*: Changes from the changed ontology are propagated to dependent ontologies only at their explicit request. The pull-based approach is better suited for less stringent consistency requirements.

According to [2], another name for change notification in change implementation phase is notification of the consequences of a change. The ontology engineer should however have possibilities to make such choices or even to abort the entire ontology evolution process when she realizes that it would have undesired consequences for other parts of the ontology, for dependent ontologies, for distributed instances or for existing applications. Consequently, she should be able to comprehend a list of all the changes and approve or cancel them. When the changes are approved, they are performed by successively resolving changes from the list. If changes are cancelled, the ontology should remain complete. In order to give an ontology-engineer a chance to cancel a change after it has been completely analyzed, it is necessary to separate the analysis of the user's request for the change from the final execution of this request within the ontology evolution system. Therefore, the main task of the change implementation phase in [2] is the application of changes. During this phase all changes (i.e. required and derived changes) are applied to a consistent ontology and result into a new consistent state of this ontology. The last task of the change implementation phase is to keep track about the performed changes. Communication about changes need a common understanding of a change model and of a log model. The evolution log tracks the history of applied ontology changes as an order sequence of information (defined through the evolution ontology) about particular change.

In [16], there are four techniques in change validation: formal verification, testing, debugging and quality assurance. Formal verification relies on formalisms such as state machines and temporal logics to derive useful properties of the system under study. Well known techniques for formal verification are model checking and theorem proving. While formal verification can be very useful, it is a technique requiring considerable expertise, and it does not always scale very well in practice. Therefore, other more pragmatic approaches are needed as well. Testing is one of these approaches. For a well-chosen representative subset of the system under consideration, tests are written to verify whether the system behaves as expected. Debugging is the task of localizing and repairing errors (that may have been found during formal verification or testing). The final activity is quality assurance, activity to ensure that the developed system satisfies all desired qualities.

Application of ontology evolution has involved various tools. Some tools that have been used in ontology evolution process are Protégé [25] & [26], OilEd [27], KAON [28], Diligent [29], DOGMA-MESS [30], Evolva [31] & [32] Ontology Evolution Visualization [33] and OntoAMAS [34]. A further explanation of utilization comparison of those tools will be discussed in the next paper.

5. CONCLUSION

There are various variations of the ontology evolution stages that have resulted from many research of late. We have shown a variety of approaches that concern the evolution process, methods and tools. Concerning future work, we plan to study the comparison of utilization of tools used in ontology evolution process.

6. REFERENCES

[1] A. M. Khattak, R. Batool, Z. Pervez, A. M. Khan, and S. Lee, "Ontology evolution and challenges," *J. Inf. Sci. Eng.*, vol. 29, no. 5, pp. 851–871, 2013.

[2] L. Stojanovic, "Methods and tools for ontology evolution," *Univ. Karlsruhe*, no. August, 2004.

[3] B. T. Gruber, "What is an Ontology ?," pp. 1–11, 1993.

[4] B. Chandrasekaran, J. R. Josephson, and V. R. Benjamins, "What Are Ontologies , and Why Do We Need Them ?," 1999.

[5] S. Editors, P. Bernus, and M. J. Shaw, *International Handbooks on Information Systems*. 2007.

[6] S. Bloehdorn, P. Haase, Y. Sure, and J. Voelker, "Ontology Evolution," *Semant. Web Technol. Trends Res. Ontol. Syst.*, pp. 51–70, 2006.

[7] P. Plessers, O. De Troyer, and S. Casteleyn, "Understanding ontology evolution: A change detection approach," *Web Semant.*, vol. 5, no. 1, pp. 39–49, 2007.

[8] P. Haase and L. Stojanovic, "Consistent evolution of owl ontologies," *Semant. Web Res. Appl.*, pp. 182–197, 2005.

[9] F. Zablith *et al.*, "Ontology evolution: a process-centric survey," *Knowl. Eng. Rev.*, vol. 30, no. 1, pp. 45–75, 2013.

[10] D. Slezak, *Ontology Evolution : A Survey and Future Challenges*. 2009.

[11] L. Stojanovic, A. Maedche, B. Motik, and N. Stojanovic, "User-driven ontology evolution management," *Knowl. Eng. Knowl. Manag. Ontol. Semant. Web*, pp. 133–140, 2002.

[12] R. Djedidi and M. A. Aufare, "Ontology Evolution: State of the Art and Future Directions," 2010.

[13] N. F. Noy and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," *Stanford Knowl. Syst. Lab.*, p. 25, 2001.

[14] A. Maedche, B. Motik, and L. Stojanovic, "Managing multiple and distributed ontologies on the Semantic Web," *VLDB J.*, vol. 12, pp. 286–302, 2003.

[15] S. Y. Haase Peter, "State of the Art on Ontology evolution," *Semant. Web Technol.*, pp. 51–70, 2004.

[16] R. Djedidi, M. A. Aufare, P. De Leenheer, and T. Mens, "Ontology Evolution: State of the Art and Future Directions," *Ontol. Theory Manag. Des. Tools Model.*, vol. 2, no. 1, pp. 1–47, 2008.

[17] P. Raul, F. Zablith, P. Haase, and O. Corcho, "Ontology Evolution.pdf." 2012.

[18] M. Klein and N. F. Noy, "A Component-Based Framework for Ontology Evolution," *Proc. 7th Int. Conf. Princ. Knowl. Represent. Reason.*, pp. 135–144, 2003.

[19] M. Klein, *Change Management for Distributed Ontologies*. 2004.

[20] O. D. Atanas Kiryakov, "Tracking Changes in RDF(S) Repositories," *Artif. Intell.*, no. July, 2002.

[21] B. Nuseibeh, S. Easterbrook, and A. Russo, "Leveraging inconsistency in software development," *Computer (Long. Beach. Calif).*, vol. 33, no. 4, pp. 24–29, 2000.

[22] A. K. Sari and W. Rahayu, "Ontology-based Change Propagation in Shareable Health Information Appications.pdf." 2015.

[23] A. K. Sari, W. Rahayu, and M. Bhatt, "An approach for

sub-ontology evolution in a distributed health care enterprise," *Inf. Syst.*, vol. 38, no. 5, pp. 727–744, 2013.

[24] F. Zablith, M. D'Aquin, M. Sabou, and E. Motta, "Using ontological contexts to assess the relevance of statements in ontology evolution," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6317 LNAI, pp. 226–240, 2010.

[25] N. Guarino and C. Welty, *Knowledge Engineering and Knowledge Management Methods, Models, and Tools*. 2000.

[26] N. F. Noy, A. Chugh, W. Liu, and M. a Musen, "A Framework for Ontology Evolution in Collaborative Environments," *ISWC'06 Proc. 5th Int. Conf. Semant. Web*, pp. 544–558, 2006.

[27] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens, "OilEd : a Reason-able Ontology Editor for the Semantic Web," *Proc. KI2001, Jt. Ger. Conf. Artif. Intell.*, pp. 396–408, 2001.

[28] T. Gabel, Y. Sure, and J. Voelker, "KAON – Ontology Management Infrastructure," *SEKT informal Deliv.*, 2004.

[29] D. Vrandecic, S. Pinto, C. Tempich, and Y. Sure, "The DILIGENT knowledge processes," *J. Knowl. Manag.*, vol. 9, no. 5, pp. 85–96, 2005.

[30] P. De Leenheer and C. Debruyne, "DOGMA-MESS: A Tool for Fact-Oriented Collaborative Ontology Evolution.," *OTM Work.*, pp. 797–806, 2008.

[31] F. Zablith, M. Sabou, M. Aquin, and E. Motta, "Ontology Evolution with Evolva Underlying Approach : The Evolva Framework," *Evolution (N. Y).*, pp. 908–912, 2009.

[32] F. Zablith, "LNCS 5554 - Evolva: A Comprehensive Approach to Ontology Evolution," pp. 944–948, 2009.

[33] P. Lambrix, Z. Dragisic, V. Ivanova, and C. Anslow, "Visualization for ontology evolution," *CEUR Workshop Proc.*, vol. 1704, pp. 54–67, 2016.

[34] S. Benomrane, Z. Sellami, and M. Ben Ayed, "An ontologist feedback driven ontology evolution with an adaptive multi-agent system," *Adv. Eng. Informatics*, vol. 30, no. 3, pp. 337–353, 2016.