An OWL-Based Ontology Structure for representing Multimodel Process Improvement Framework

Abdel Nasser H. Zaied
Professor of Information Systems, College of Computer Science, Misr International University, Egypt
Abdelnasser.riad@miuegypt.edu.eg, nasserhr@gmail.com

Khalid A. Eldrandaly
Dean, College of Computers and Informatics, Zagazig University, Egypt
khalid_eldrandaly@zu.edu.eg, khalid_eldrandaly@yahoo.com

AlShaimaa A. Tantawy
Lecturer, Department of Information Systems, College of Computers and Informatics, Zagazig University, Egypt
AlshaimaaTantawy@zu.edu.eg, Tantawyalshaimaa@gmail.com

Abstract- *Software and systems improvement requests to merge various interpretations from several improvement models and techniques. A particular challenge is the multitude of models for requirements and quality, which can get time consuming and error prone to trace, change, and verify. Lately, Ontologies have been used across several domains and for numerous purposes to be applied for many applications. Besides, recent work in Artificial Intelligence is discovering the use of formal ontologies as a way of identifying content-specific agreements for the sharing and reuse of knowledge among software entities. Therefore, this paper describes how ontology engineering is used to construct an Ontological structure of the proposed SPI-CMMI framework –which based on using Six sigma approach integrated with CMMI-Dev model and Quality Function Deployment (QFD) technique- with its progressive phases, related activities, recommended tools and the CMMI-Dev 1.3 representation. The SPI-CMMI Ontology provides a shared improvement terminology, defines precise and unambiguous semantics for the software enterprises and enables reuse of improvement phase's knowledge; in addition it makes domain assumptions explicit and separate domain knowledge from the operational knowledge.*

Keywords: *CMMI-Dev model, Multi-model environments, Ontology engineering, OWL, Six Sigma, Software Process Improvement.*

1. Introduction

The term *ontology* has its origin in philosophy, and has been applied in several different behaviors. The word "Ontology" comes from the Greek *Ov* (on), which literally means entity. The core meaning within computer science is a model for describing the world that consists of a set of types, properties, and relationship types. Ontology in common is the representation of entities, ideas, and events, along with their properties and relationships, according to a system of categories [12]. Ontologies aim to capture consensual information and knowledge within their relationships in a generic and formal way, and that they may be reused and shared across other applications (or software programs) and by groups of people in different locations for various purposes [1]. Contemporary ontologies share many structural similarities, regardless of the language in which they are expressed. Most Ontologies describe individuals (instances), classes (concepts), attributes, relations, function terms, restrictions, rules, events, and axioms. Ontology languages are usually declarative languages, generalizations of frame languages, and based on either first-order logic or on description logic. There are a number of such languages for the ontologies' representation, both proprietary and standards-based, *such as*; OBO, Common Algebraic Specification Language, IDEF5, DOGMA, Web Ontology Language (OWL), Rule Interchange Format (RIF), SADL ...etc. [12].

The available international models, methodologies and techniques for Software Process Improvement (SPI) can be classified into two paradigms; the benchmark and the analytical based process improvement approaches. Benchmark based approaches are prescriptive in nature, defining requirements or advising a set of practices originating from top performing organizations, that are adopted by organizations aiming to improve their software process. Analytical approaches are based on strategies that aim first, to define business, process and product objectives and then establish a clear understating of the impact of process performance in these objectives. A recent trend in SPI is the adoption of more than one improvement model into a single organizational environment, originating what are denominated multi-model environments. The goal is

to achieve the cumulative added benefit of adopted models [3].

With the aim of enhancing the capabilities of the SPI models, especially the CMMI-Dev 1.3 model; [20] adopted the improvement multi-model environment through proposing a practical improvement methodology which is called the proposed SPI-CMMI framework that is based on integrating the Six sigma approach (analytical toolkits, techniques and methodology) and Quality Function Deployment (QFD) technique within the CMMI-Dev v1.3 model. The proposed systematic methodology helps the organization to optimize and improve the existing processes in addition to facilitating the adoption process of the CMMI-Dev model, thus it is named the proposed SPI-CMMI framework.

The combination significance is the mixture of their best practices in a comprehensive improvement strategy, that otherwise would not be possible to obtain by a single technological approach. Therefore, the SPI-CMMI framework fills in "*what/how/why*" technologies combination which provides theoretically what improvement processes should be done to satisfy most of the critical stakeholders' requirements and practically how the organization and improvement processes can be executed efficiently using the analytical toolkits, appropriate techniques, and the detailed steps or action plans.

The purpose of the work presented in this article is the development of an ontological structure of the SPI-CMMI framework proposed in [20] for the system and software improvement; As Ontologies engineering are accepted in this research as means for enabling the software organization, representing, storage, querying and retrieval of knowledge used in the SPI-CMMI in an organizational memory system. They do so by defining a common understanding or vocabulary between people and across a range of applications. When models/standards are presented in ontology, they gain the abilities of machine process ability, share ability, and querying. Liao et al. [10] presented the advantages of ontology use for process modeling clearly. In addition, when both of process reference models/standards and organizational processes are represented by ontologies, they can share the same concepts, be mapped to each other and queried.

This paper is organized as follows. Section 2 presents briefly the SPI-CMMI framework. In Section 3, the related Literature Survey is displayed. The structure of SPI-CMMI ontology is introduced in details in Section 4. Finally, some conclusions and future work are drawn in Section 5.

3 The SPI-CMMI Framework

The proposed SPI-CMMI framework will show how to use Six Sigma methodology, toolkits, metrics and QFD to meet CMMI-DEV v1.3 guidelines, to incrementally improve the maturity of the software development organization. It targets all companies that develop software and seeking to make improvements within their current software development process using CMMI. The SPI-CMMI framework contains ten phases illustrated briefly as the following:

Phase-1: Improvement Project Initiation using Six Sigma tools and metrics to evaluate the organization current state, through determining the capabilities, strengths, and weaknesses to specify where the organization should start the improvement process.

Phase-2: Performance Management and Success Metrics Derivation

It is very important for any process improvement effort to determine which measures should be specified in order to show improvement progress and benefits. An important methodology for deriving success metrics is the Goal-Question-Metric (GQM) approach and the Dashboard document to track and record the metrics.

Phase-3: Requirements Collection and Prioritization Collecting the requirements from all the stakeholders, developing a method based on QFD and the priority assessment technique for the integration and prioritization of requirements from multiple perspectives; Customer, Business, Management, Quality …etc.

Phase-4: CMMI Process Areas Prioritization

For each of the process categories in the CMMI continuous representation (or for each maturity level in the CMMI staged representation), the set of requirements with adjusted priorities are related to the specific PAs. The specific PAs are prioritized based on those process requirements. Thus, the PAs that achieve higher overall satisfaction of process requirements get higher importance.

Phase-5: CMMI-Dev Specific Goal Prioritization

For each prioritized process area, the set of requirements with adjusted priorities are related to the specific goals. The specific goals are prioritized based on those process requirements. Thus, the specific goals that achieve higher overall satisfaction of process requirements get higher importance.

Phase-6: Specific Practices Prioritization involves the prioritization of Specific Practices within all PAs of a specific maturity level (Staged CMMI) or within all PAs of a specific process category (Continuous CMMI). The prioritization is carried out on the basis of the deliverables from Phase 5. According to CMMI specifications, all these Specific Practices

have to be performed to reach that particular maturity level.

Phase-7: Action Plans Derivation and Prioritization, a set of actions is derived from the prioritized practices. The priorities of actions reflect the priorities of process requirements. By executing the actions with the highest priorities, the highest satisfaction level of process requirements can be achieved.

Phase-8: Action Plans and Practices Implementation. Using the appropriate Six sigma tools, methods, techniques and suggested metrics in applying the prioritized practices and action plans for each process area, in order to ensure much more successful implementation of the organization's CMMI specific goals and practices in accurate and fast manner.

Phase-9: CMMI Capability Levels Interpretation. Process capability deals with the how well defined and managed the process is. Generic goals and practices are those activities that ensure that the process improvements identified will be effective over the long term. They should be implemented to all of the process areas within the CMMI. This phase includes suggested activities and steps within the six sigma methodology (DMAIC).

Phase-10: Capability Levels Activities Implementation. Using the appropriate Six sigma tools, methods, techniques and suggested metrics in applying the suggested activities and steps for each capability level, in order to ensure much more successful implementation of the organization's CMMI generic goals and practices in precise and managed manner.

Figure 1 summarizes the main progressive steps suggested to be applied within the SPI-CMMI framework in the software enterprise.

3. Literature Survey

There are only limited studies on CMMI ontology in the literature, illustrated in the succeeding pieces.

Liao et al. 2005 produced an OWL-based ontology for generic Software Process (SPO) and attempted to ensure that it covered the requirements of both CMMI and ISO/IEC 15504. His study indicated that an organization's process model could be represented by using SPO and that a web-based process assessment tool that used SPO has been under development [10].

Soydan, and Kokar 2006 provided a short description of Ontology for CMMI-SW. The ontology was coded in a formal language, OWL. Some test cases were used to assess the ontology validity by means of an OWL reasoner to derive the results [18]. Only staged representation was analyzed whereas in this research, it is designed to meet the requirements of both staged and continuous representations.

Rungratri and Usanavasin 2008 proposed a framework called "CMMI v1.2 based Gap Analysis Assistant Framework (CMMI-GAAF)" to perform automatic gap analysis with respect to CMMI. Also, Project Assets Ontology (PAO) was created based on CMMI ontology developed in [18] to merge CMMI process areas and project assets [16].

Ferchichi et al. 2008 applied ontology to the integration of ISO 9001:2000 and CMMI to generate a multi-vues quality ontology allowing a double certification relative to these two standards. This work was especially carried out only within a software engineering company (Sylis) [2].

Lee et al. 2008 proposed an ontology-based intelligent estimation agent, including a CMMI-based project planning ontology and a fuzzy cost estimation mechanism, for the total project cost estimation. Based on the information stored in the CMMI-based project planning ontology predefined by domain experts, the fuzzy cost estimation mechanism inferred the total project cost and then stored the related results to the project estimation repository [9].

Sharifloo, et al. 2008 introduced an ontology system to represent the CMMI-ACQ v1.2 domain knowledge. This ontology has been developed based on Suggested Upper Merged Ontology (SUMO) using SOU-KIF languages [17].

Lee et al. 2008 presented an ontology-based intelligent decision support agent (OIDSA) to apply to project monitoring and control of CMMI. The OIDSA was composed of a natural language processing agent, a fuzzy inference agent, and a performance decision support agent. The OIDSA could be work for only project monitoring and control of CMMI [8].

Lee and Wang 2009 presented fan ontology-based computational intelligent mutli-agent for CMMI assessment. The system comprised a natural language processing agent, an ontological reasoning agent, and a summary agent to summarize the evaluation reports. It was built based on process and product quality assurance process area of CMMI [7].

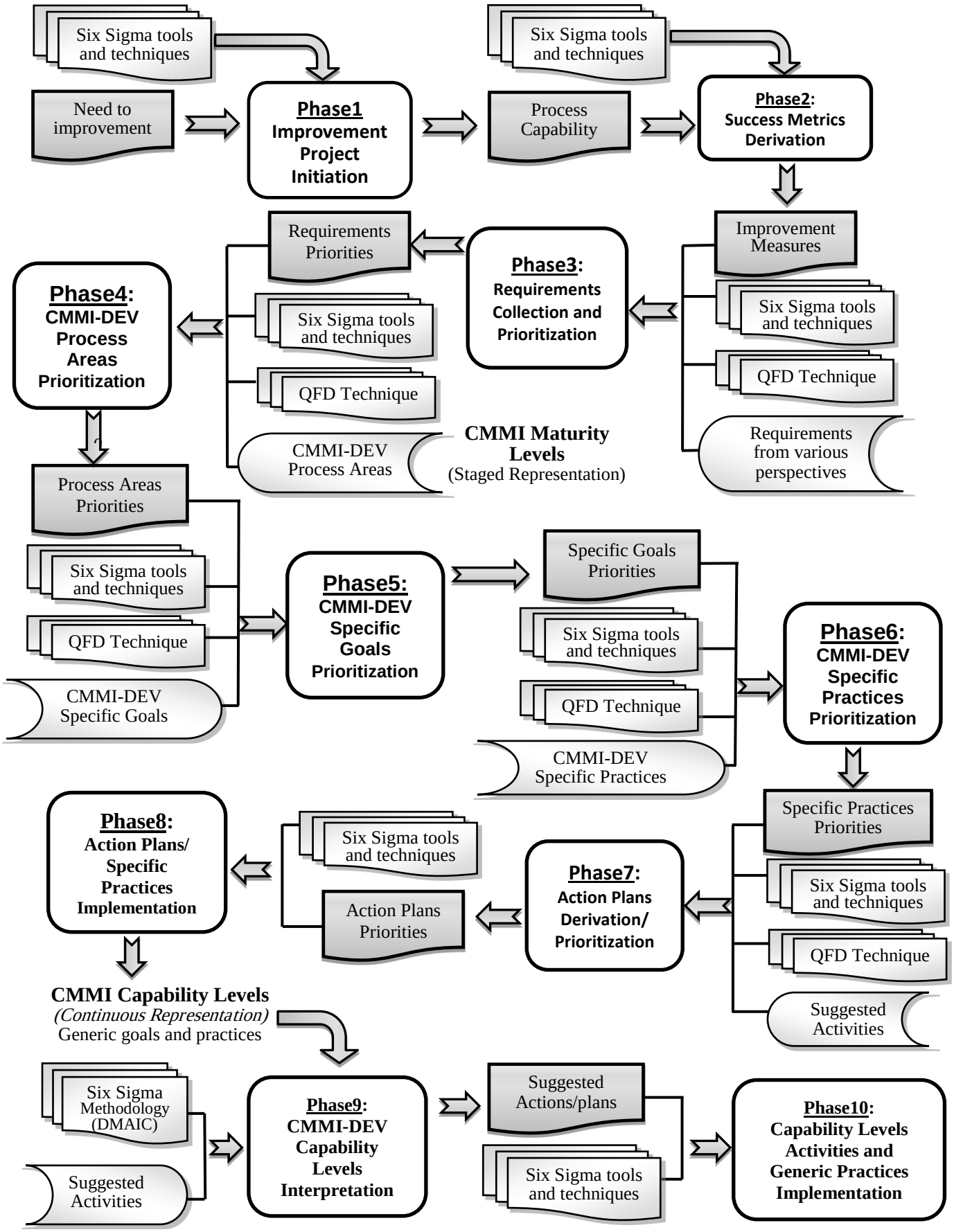


Figure 1: The illustration of the SPI-CMMI phases and steps.

Pardo et al. 2012 presented ontology for the harmonization of multiple models. It was supported by a web tool and; had been applied for the harmonization of COBIT 4.1, Basel II, VAL IT, RISK IT, ISO 27002 and ITIL [13].

Soydan, and Kokar 2012 presented a formalization of CMMI-Dev model. The formalization was expressed in OWL. This formalization aimed to be consistent with CMMI-Dev and to be operational, *i.e.*, to allow for an automatic determination of a development process maturity level based upon data about the practices within a given organization. For the formalization validity, a number of test cases for the scenario of automatic determination of the maturity level were developed [19].

Gazel et al. 2012 developed an ontology-based SPA tool to support data collection phase of process assessment and to track conformance of software processes to CMMI as the process reference model. Ontology-based CMMI Mapping and Querying Tool (OCMQT) was developed as a plug-in to an open-source process management tool, namely EPF Composer which, was a realization of the process engineering meta-model SPEM [4].

Mejia et al. (2016) presented an ontological framework based on a multi-model approach, which facilitates and supports the SPI for small and medium companies for a life cycle process improvement. They presented a case study to show the performance of the framework [11].

4 The Structure of SPI-CMMI Ontology

4.1 OWL Ontology Language

The OWL ontology language from the World Wide Web Consortium (W3C), with the Protégé editor is selected for constructing the SPI-CMMI Ontology according to the following reasons [6]:

- OWL language makes it possible to describe concepts and provides new facilities. It has a richer set of operators - e.g. intersection, union and negation. It is based on a different logical model which makes it possible for concepts to be defined as well as described. Complex concepts can therefore be built up in definitions out of simpler concepts.
- Furthermore, the logical model allows the use of a reasoner which can check whether or not all of the statements and definitions in the ontology are mutually consistent and can also recognize which concepts fit under which definitions.

Subsequently, for the construction of the SPI-CMMI ontology; the Protégé v3.5 (Build 663) is selected as an ontology editor and knowledge-base framework. Protégé is developed at the Stanford Center for Biomedical Informatics Research (BMIR) at the Stanford University School of Medicine. The Protégé editor provides the successive facilities [15]:

- Protégé is a free, open-source platform that provides a growing user community with a suite of tools to construct domain models and knowledge-based applications with ontologies.
- Protégé can be customized to provide domain-friendly support for creating knowledge models and entering data that is because the Protégé platform supports modeling ontologies via a web client or a desktop client.
- Protégé implements a rich set of knowledge-modeling structures and actions that support the creation, visualization, and manipulation of ontologies in various representation formats.
- Protégé ontologies can be developed in a variety of formats including OWL, RDF(S), and XML Schema.
- Protégé is based on Java, is extensible, and provides a plug-and-play environment that makes it a flexible base for rapid prototyping and application development.
- Protégé is supported by a strong community of developers and academic, government and corporate users, who are using Protégé for knowledge solutions in areas as diverse as biomedicine, intelligence gathering, and corporate modeling.

The Protégé platform supports two main ways of modeling ontologies [4]:

- **The Protégé-Frames editor** enables users to build and populate ontologies that are *frame-based*, in accordance with the Open Knowledge Base Connectivity protocol (OKBC). In this model, ontology consists of a set of classes organized in a hierarchy to represent a domain's salient concepts, a set of slots associated to classes to describe their relationships, and a set of instances of those classes - individual exemplars of the concepts that hold specific values for their properties.
- **The Protégé-OWL editor** enables users to build ontologies for the *Semantic Web*, in particular in the W3C's Web Ontology Language (OWL). "OWL ontology may include descriptions of classes, properties and their instances. Given such ontology, the OWL formal semantics specifies how to derive its logical consequences.

The Ontology representation of the proposed SPI-CMMI framework is constructed briefly according to the subsequent steps:

- First, the proposed SPI-CMMI framework is initially formalized as ontology that captures the main concepts and properties of the SPI-CMMI framework, according to the interpretation of the ontology perception that is used in knowledge representation. It is called the SPI-CMMI Ontology.
- This ontology is then used to represent the ten phases with their suggested activities of the SPI-CMMI framework, QFD technique and the six sigma approach with the full formalization of the CMMI-Dev v1.3 model with its two representations staged and continuous.
- In the next step, a generic OWL reasoner, the built-in reasoners in protégé, Pellet1.5.2 reasoner [14], is used to verify the consistency checking of the representation, concept satisfiability, classification, and realization.

4.2 Naming Conventions in OWL Ontology

The naming style followed in constructing the SPI-CMMI ontology is capitalization of class names; *for example;* Phase_One, Success_Metrics, Maturity_ Levels, and ProcessArea_ML4, and object properties names with low-case letters. For the Object Properties (Relations connects between classes) in SPI-CMMI ontology, the recommended style of using an action verb as a prefix to the property; such as executed_By, consists_Of, implemented_By, and has_Precedence. Figure 2 illustrates the detailed formalization of the SPI-CMMI ontology structure.

In OWL ontology, everything is a subclass of owl:Thing. So the SPI-CMMI Framework class is defined as subclasses of owl:Thing, and has 13 subclasses belongs to it: Phase One, Phase Two, Phase Three, Phase Four, Phase Five, Phase Six, Phase Seven, Phase Eight, Phase Nine, Phase Ten, CMMI-DEV_1.3 Model, Six Sigma Tools, QFD Technique.

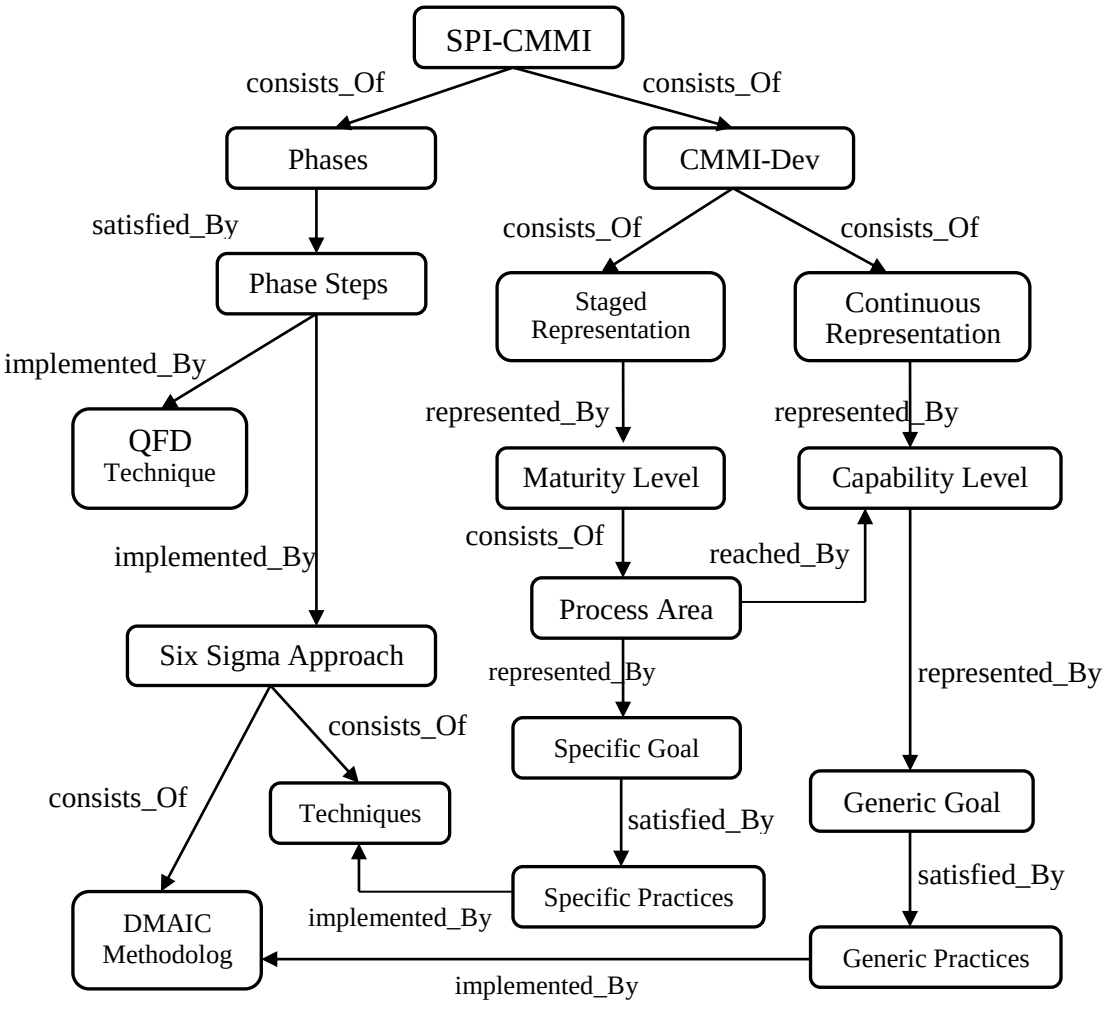


Figure 2: The detailed formalization of the SPI-CMMI ontology structure

Each subclass has its own subclasses; *for instance*, Six Sigma Tools class has three subclasses: Methodologies, Techniques, and Graphical Methods, and so on. Figure 3 provides an instant of the class hierarchy of the SPI-CMMI ontology using Protégé editor v3.5 (Build 663).

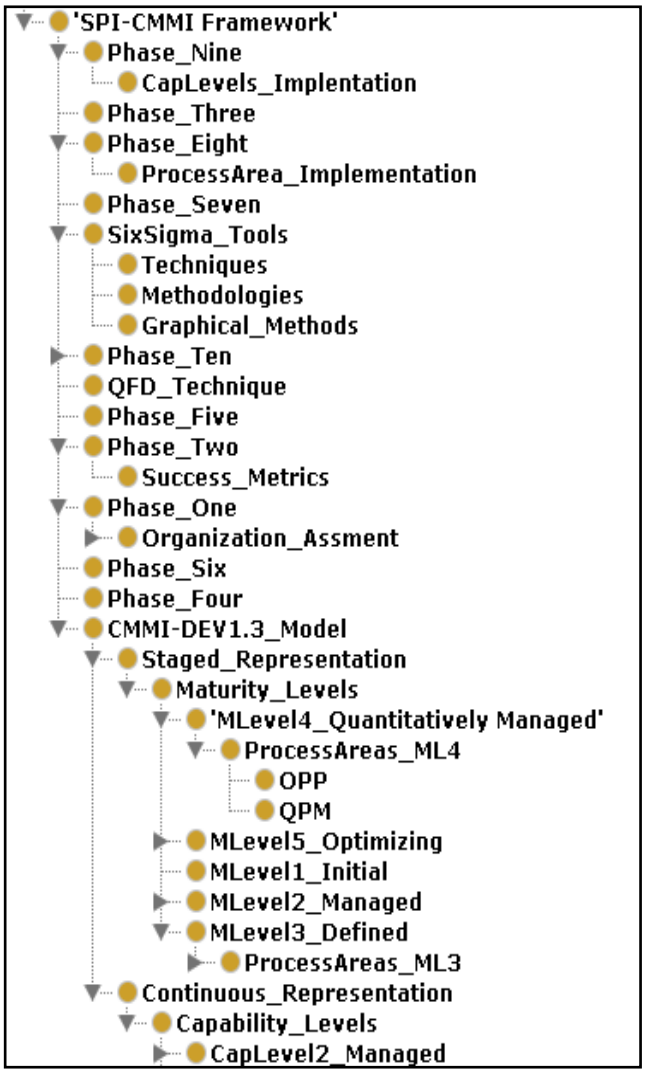


Figure 3: The Class Hierarchy of SPI-CMMI Ontology

Here is a code sample as an example of the class definitions for the CMMI-Dev 1.3 Model class:

```
<owl:Class rdf:ID="CMMI-DEV1.3_Model">
    <rdfs:subClassOf rdf:resource="#SPI-CMMI_Framework"/>
    <rdfs:comment rdf:datatype="&xsd;string"
      >CMMI-DEV v1.3 consists of best practices that address development activities applied to products and services. It addresses practices that cover the Product & # 8217; s lifecycle from conception through delivery and maintenance.</rdfs:comment>
    <rdfs:label rdf:datatype="&xsd;string"
      >CMMI-DEV1.3_Model</rdfs:label>
  </owl:Class>
```

There are two main kinds of properties (relations) in OWL Ontology: Object Properties and Data type Properties.

- The first category is the Object Properties (relations connect between classes) which link an individual to another class or individual, e.g. consists_Of - executed_By - has_Precedence - implemented_By.
- The second category of properties in OWL Ontology is the Data type Properties that link an individual to a precise value. As an illustration, the following is the OWL-Datatype Property Definition for phase_description.

For the SPI-CMMI ontology representation using Protégé editor, Table 1 provides a brief explanation of object properties in the SPI-CMMI ontology.

Table 1: Object Properties in the SPI-CMMI Ontology

Name	Concepts	Descriptions
contain	Staged Representation → Maturity Levels Continuous Representation → Capability Levels	Return capability/ maturity levels that exist in continuous /staged.
consists_Of	Maturity Level →Process Area	Reflects all the PAs that require for achieving level.
executed_By	Generic Practices → DMAIC Activities	Returns DMAIC phases/ activities that execute the capability levels.
guarante_By	Process Area → Suggested Metrics	Represent the suggested metrics for each PA.
has_ Precedence	SPI Phases with each other CMMI Maturity Levels CMMI Capability Levels	Reflects all previous phases that require to be satisfied first.
Implemented _By	Assessment Steps → Six Sigma Tools Success Metrics →Six Sigma Tools SPI Phases → Sigma Tools SPI Phases →QFD Specific Practices → Six Sigma Tools	Represents recommended tools, suggested activities and techniques for implementing the SPI phase, CMMI level, and PA.
reached_By	Generic Goals→ Generic Practices SpecificGoals→Specific Practices	Reflects the specific practices needed to reach the specific goals.
represented_By	CMMI-DEV 1.3 Model → Staged Representation CMMI-DEV 1.3 Model → Continuous Representation	Reflects the continuous and staged in CMMI model.

As the CMMI-Dev model is considered the main class (object) in the SPI-CMMI ontology structure, Figure 4 revealed an extract of the instances of the CMMI-Dev representation using Protégé.

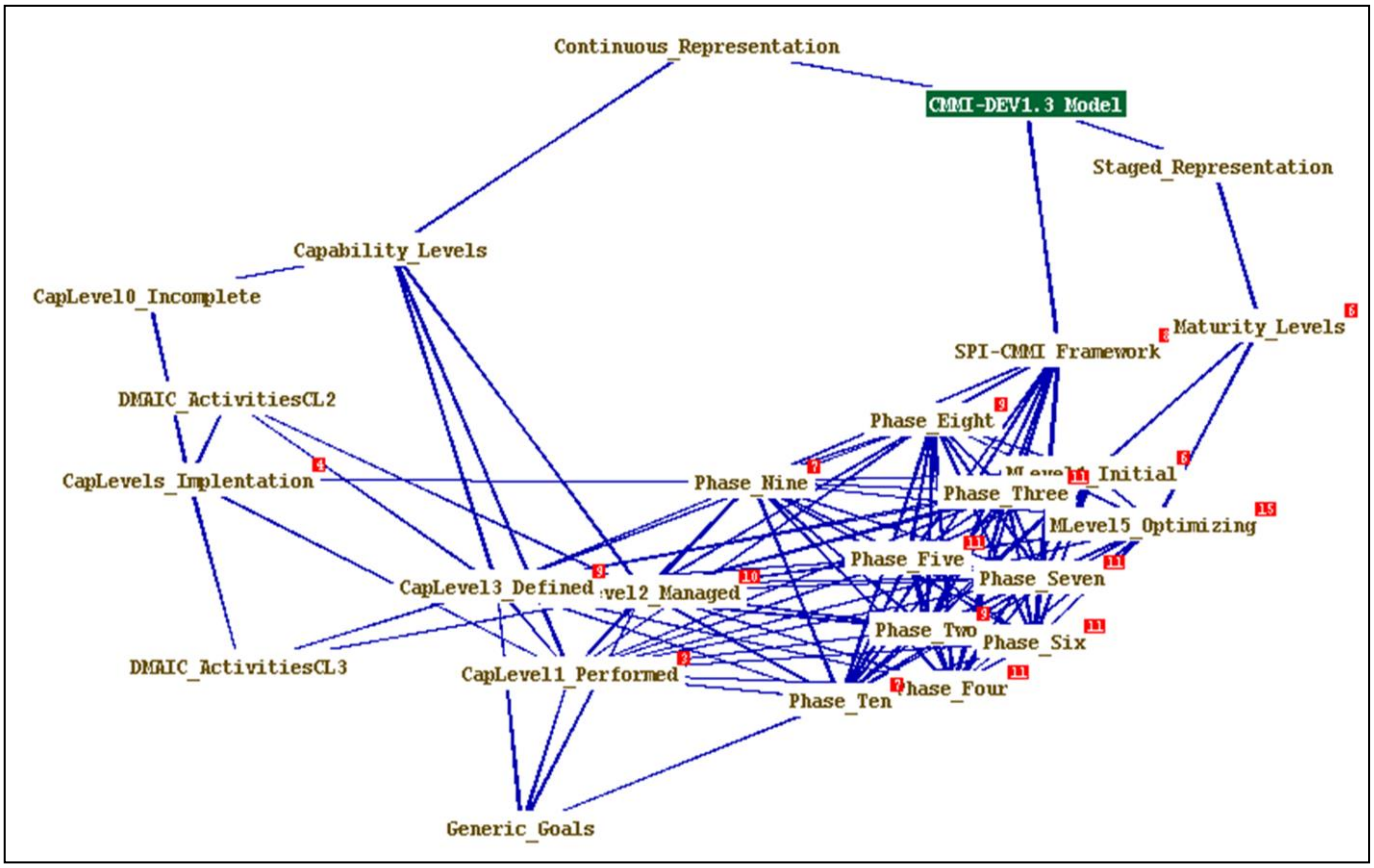


Figure 4: The CMMI-Dev Representation in SPI-CMMI Ontology using Protégé editor

As stated previously, Protégé is based on Java; the following code sample gives an example of Java code generated by Protégé editor for the class of the six sigma tools in the SPI-CMMI ontology structure.

```
import java.util.*;
/**
 * Generated by Protege (http://protege.stanford.edu).
 * Source Class: SixSigma_Tools
 *
 * @version generated on Thu Aug 17 12:44:58 EEST 2017
 */
public interface SixSigma_Tools extends
SPI_CMMI_Framework {
   // Slot tool_Description
   Collection<String> getTool_Description();
   boolean hasTool_Description();
   void addTool_Description(String newTool_Description);
   void removeTool_Description(String oldTool_Description);
   void setTool_Description(Collection<String>
newTool_Description);
   // Slot tool_Name
   String getTool_Name();
   boolean hasTool_Name();
   void setTool_Name(String newTool_Name);
   void delete();
}
```

4.3 Design Criteria for SPI-CMMI ontology

Five main objective criteria for designing ontologies were established by Gruber [5] to guide and evaluate the ontologies designs whose specific intention is knowledge sharing. A brief summarize of how the SPI-CMMI ontology has taken them into consideration will be given in the following illustration. The five criteria are:

- **Clarity**: Ontology should use objective definitions that are as complete as possible. The complete definitions of the SPI-CMMI ontology are documented with natural language.
- **Coherence**: Inferences in ontology should be consistent with the definitions. The defining axioms should be logically consistent. This research is focused on representation, retrieval, and query rather than reasoning, while reasoning within the SPI domain might be a possibility. Because reasoning is not the intent of this research, no deliberate inferences were made in the design of the SPI-CMMI ontology. However, logical relationships can be implied by the structure and definitions of ontology. To check consistency, the SPI-CMMI ontology is tested by reasoning software as part of the later implementation process.

- **Extendibility**: Ontology should be designed to anticipate the uses of the shared vocabulary. It should offer a conceptual foundation for a range of anticipated tasks. The SPI-CMMI ontology can be easily extended and specialized for each organization, through adding new phases, activities or tools when needed without having to revise the existing definitions.
- **Minimal Encoding Bias**: The representation format or language for the ontology should not introduce constraints that are caused by the language and not the ontology. This was actually achieved with the SPI-CMMI ontology by explicitly designing the ontology structure before the choice of a particular knowledge representation was made. Only then were the available encodings considered.
- **Minimal Ontological Commitment**: Ontology should make the least number of assertions regarding the domain being modeled that will still enable knowledge transfer. This was the philosophy in choosing the knowledge attributes for the representation of the SPI-CMMI ontology.

4.4 The SPI-CMMI Ontology Reasoners

Although the SPI-CMMI ontology was designed according to Gruber's guidelines and criteria, there is an empirical check that can be performed on the SPI-CMMI ontology. One of the benefits of the OWL based ontologies is that they can be processed by a reasoner for consistency, species, and inferences. Consistency checking ensures that no class is defined such that it cannot have a logical instance. Species validation determines the sub-language of OWL that is being employed by the ontology. Several reasoners are available for OWL, including Hermit, FaCT, and Pellet.

Because of its seamless integration with Protégé, Pellet reasoner [13] was chosen as the tool for consistency checking, classification and.

After running the SPI-CMMI ontology through Pellet 1.5.2 reasoner, it was determined that there were no inconsistencies as described below in Figure 5.

Also, Figure 6 shows the classification of the SPI-CMMI Ontology using the Pellet 1.5.2 reasoner.

The inferred OWL sub-language for the SPI-CMMI ontology representation is OWL-DL. The computing inferred types of the SPI-CMMI ontology is presented below in Figure 7.

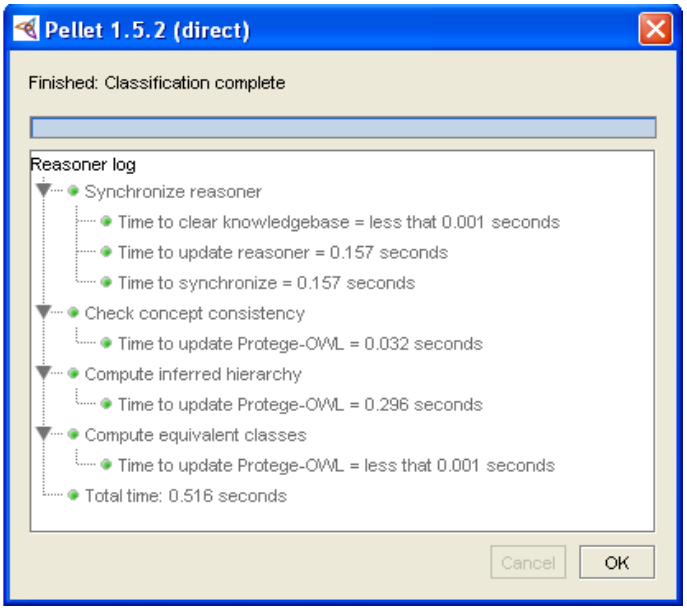


Figure 5: Classification of the SPI-CMMI Ontology

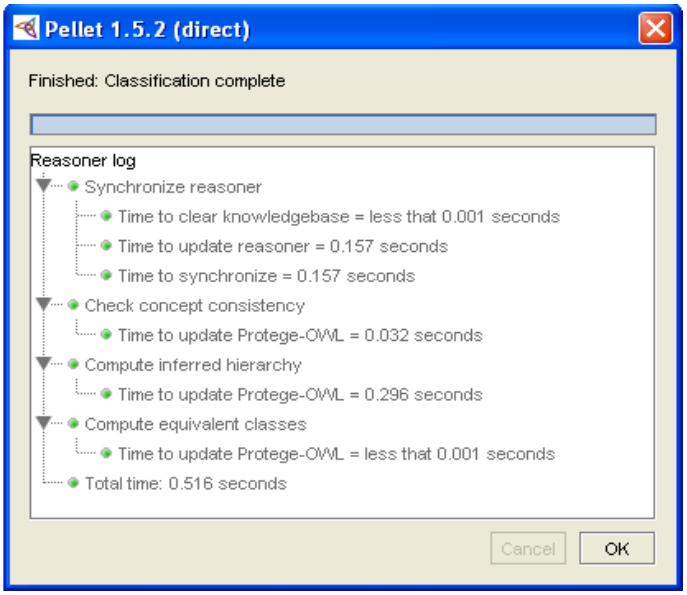


Figure 6: Classification of the SPI-CMMI Ontology

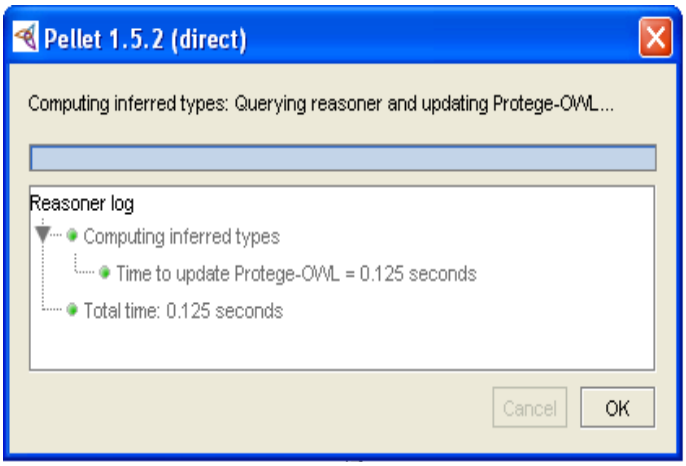


Figure 7: Computing inferred types of the SPI-CMMI Ontology

4.5 Benefits of the SPI-CMMI Ontology

The ontology construction for the SPI-CMMI framework enables improvement semantic and knowledge capture, sharing and retrieval through establishing a common conceptualization. Aligning it with various SPI processes makes relevant knowledge capture and retrieval more probable because it will happen relative to a particular improvement phase. Using a standard representation language to implement this SPI-CMMI ontology can ease implementation of the proposed SPI-CMMI framework in the software development enterprises and make it adaptable to future uses and changes.
Developing the SPI-CMMI ontology is similar to defining a set of software improvement data and their relationships in a rigorous structure for other improvement applications or programs to use. As, Problem-solving methods, domain-independent applications, and software agents use ontologies and knowledge bases built from ontologies as data.
The SPI-CMMI ontology provides the following benefits:

- Assist the software enterprises adopt CMMI with detailed steps and clear relations.
- The SPI-CMMI ontology defines a common vocabulary for researchers who need to share information in the SPI domain. It includes machine-interpretable definitions of basic concepts in SPA, SPI, CMMI, Six Sigma tools and relationships among them.
- The software enterprises can use the SPI-CMMI ontology throughout improvement knowledge practice, retrieval and query.
- The SPI-CMMI ontology enables sharing common understanding of the structure of SPI information among people or different software agents.
- It enables reuse of improvement phase's knowledge; in addition it makes domain assumptions explicit and separate domain knowledge from the operational knowledge.
- The SPI-CMMI ontology provides a complete structure and analysis of the knowledge represented in the SPI-CMMI framework with its suggested phases, related detailed steps, activities, toolsets and recommended techniques.
- It could be cheaper, faster and easier to be learned, trained, and used; also, it decreases the probable mistakes and confusion.
- It offers a clear understanding of the relationships between the SPI phases, numerous Six Sigma tools, CMMI practices, goals and process areas in an organization.
- Moreover, any modification in the improvement phases/activities or CMMI-Dev model could be relatively easily applied, and a new version could be made available to the users in a moderately short time.

5. Conclusion and Future Work

The main purpose of this implementation in this paper is the construction of the ontological structure for the proposed SPI-CMMI framework with the OWL Ontology language using the protégé editor. The SPI-CMMI Ontology can support the requirements engineering and SPI processes through providing an interrelated model for software-intensive systems, their environment, and improvement processes supporting elicitation, representation, and analysis of the interdependencies among various improvement models and domain levels. For future work the SPI-CMMI Ontology can be expanded in order to add more functions, or to be integrated with other corresponding ontologies.

References

[1] Corcho, O., Fernández-López, M., and Gómez-Pérez, A., "Methodologies, tools and languages for building ontologies. Where is their meeting point?", Data & Knowledge Engineering, vol. 46, pp. 41–64, 2003.

[2] Ferchichi, A.; Bigand, M.; and Lef`ebvre, H., "An Ontology for Quality Standards Integration in Software Collaborative Projects", Proceedings of MDISIS, vol. 430, pp. 17-30, 2008.

[3] Ferreira, A. L., *"Methodological approaches for software process improvement in multi-model environments*", Ph.D. thesis, engineering school, Minho's university, 2016.

[4] Gazel, S., Sezer, E. and Tarhan, A., "An Ontology Based Infrastructure to Support CMMI Based Software Process Assessment", Gazi University, Journal of Science, vol. 25(1), pp. 155-164, 2012.

[5] Gruber, T.R., "Toward Principles for the Design of Ontologies for Knowledge Sharing", International Journal of Human-Computer Studies, vol. 43(5-6), pp. 907-928, 1995.

[6] Horridge, M., A Practical Guide to Building OWL Ontologies using Protégé 4 and CO-ODE Tools, ed. 1.3, The University of Manchester, 2011. Available at [http://130.88.198.11/tutorials/protegeowltutorial/]

[7] Lee, C.-S.; and Wang, M.-H., "Ontology-based computational intelligent multi-agent and its application to CMMI assessment", Appl. Intell., vol. 30, pp. 203–219, DOI 10.1007/s10489-007-0071-1, 2009.

[8] Lee, C.-S.; Wang, M.-H.; and Chen, J.-J., "Ontology-based intelligent decision support

agent for CMMI project monitoring and control", International Journal of Approximate Reasoning, vol. 48, pp. 62–76, 2008.

[9] Lee, C.-S.; Wang, M.-H.; Yan, Z.; Lo, C.; Chuang, H.; and Lin, Y., "Intelligent estimation agent based on CMMI ontology for project planning, Systems, Man and Cybernetics", SMC 2008. IEEE International Conference, 12-15 Oct., pp. 228–233, 978-1-4244-2384-2, Singapore, 2008.

[10] Liao, L., Qu, Y., and Leung, H., "Software Process Ontology and Its Application". In ISWC, Workshop on Semantic Web Enabled Software Engineering, 2005.

[11] Mejia, J.; Muñoz, E.; and Muñoz, M., "Reinforcing the applicability of multi-model environments for software process improvement using knowledge management", Science of Computer Programming vol. 121, pp. 3–15, 2016.

[12] Ontology Available at http://en.wikipedia.org/wiki/Ontology_(information_science) [last Access 17/8/2017]

[13] Pardo, C.; Pino, F. J.; García, F.; Piattini, M.; and Baldassarre, M. T., "An ontology for the harmonization of multiple standards and models", Computer Standards and Interfaces, vol. 34, pp. 48–59, 2012.

[14] Pellet Available at [http://pellet.owldl.com/] (Last Access 22/2/2014)

[15] Protégé Available at http://protege.stanford.edu (Last Access 17/8/2017)

[16] Rungratri, S., and Usanavasin, S., "Project Assets Ontology (PAO) to Support Gap Analysis for Organization Process Improvement Based on CMMI", Making Globally Distributed Software Development a Success Story, Springer, Berlin/Heidelberg, May, pp. 76-87, 2008.

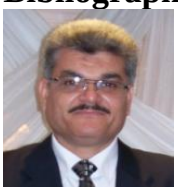
[17] Sharifloo, A. A.; Motazedi, Y.; Shamsfard, M.; and Dehkharghani, R., "An Ontology for CMMI-ACQ Model, Information and Communication Technologies: From Theory to Applications", IEEE, ICTTA. 3rd International Conference, 7-11 April, p.p. 1–6, 978-1-4244-1752-0, Damascus, 2008.

[18] Soydan, G. H., and Kokar, M. M., "An OWL Ontology for Representing the CMMI-SW Model," The 2nd International Workshop on Semantic Web Enabled Software Engineering, Athens, 6 November 2006.

[19] Soydan, G. H., and Kokar, M. M., "A Partial Formalization of the CMMI-DEV—A Capability Maturity Model for Development", *Journal of Software Engineering and Applications*, Vol 5, pp. 777-788, 2012.

[20] Zaied, A.N.H., El-Drandaly, K.A., and Tantawy, A.-S.A.: "A proposed Framework for Software Process Improvement: Extending CMMI-DEV model Using Six Sigma and Quality Function Deployment Techniques", Kafrelsheikh Journal of Information Sciences, 1, (1), pp. 21-32**,** 2018.

Bibliography



Prof. Abdel Nasser H. Zaied is a Professor of Information Systems, College of Computer Science, Misr International University, Egypt. He previously worked as a Dean, College of Computers and Informatics, Zagazig University, Egypt; as a Former Adviser to the Minister of Higher Education for Private and National Universities; as a Professor of Industrial Engineering, Zagazig University Egypt; as an Assistant Professor of Technology Management, Arabian Gulf University, Bahrain; and as Visiting Professor at Oakland University, USA. He supervised 19 PhD theses and 57 MSc. theses, and examined 60 MSc & 23 PhD theses. He published 66 research papers in International and Regional Journals and 27 research papers in International and National conferences. His areas of research are: Systems Analysis and Design; Information Security; Quality Management Systems, Project Management and Electronic applications**.**

Prof. Khalid A. El-Drandaly is a Professor of Information Systems, Dean, College of Computers and Informatics, CIO and Director of the Communication and Information Technology Center, Zagazig University, Egypt. He is a certified GIS professional (GISP). He received his Ph.D. degree in Systems Engineering (GIS). His research interests include GIS, Expert Systems, SDSS, MCDM, and Intelligent Techniques in Decision Making. He is a member of the Egyptian Engineers Syndicate, Texas A&M International Faculty Network, ESEA, GIS Certification Institute, International Society for Environmental Information Science, and ACM. He serves as a member in Review Committee of IAJIT, IJGIS, IJOPCM, ASOC, ACIT, JEI, FCT and AJSE.

Al-Shaimaa A. Tantawy is aa assistant professor of Information Systems, IS Department, College of Computers and Informatics, Zagazig University, Egypt. Her research interests include; Information System, System analysis and Design, Software Engineering, Software Quality Assurance, Software Process Improvement, and Ontology.