Journal of Software Engineering and Applications, **2012, 5, 777-788**
doi:10.4236/jsea.2012.510090 Published Online October 2012 (http://www.SciRP.org/journal/jsea)

A Partial Formalization of the CMMI-DEV—A Capability Maturity Model for Development

Gokhan Halit Soydan, Mieczyslaw M. Kokar

Department of Electrical and Computer Engineering, Northeastern University, Boston, USA.
Email: gsoydan@ece.neu.edu, mkokar@ece.neu.edu

Received August 2nd, 2012; revised September 5th, 2012; accepted September 16th, 2012

ABSTRACT

CMMI (Capability Maturity Model Integration) is a set of models—collections of best practices intended to help organizations to improve their processes. CMMI-DEV provides guidance to development organizations. This paper presents a formalization that captures definitions of a number of concepts of CMMI-DEV and relations among the concepts. The formalization is expressed in a formal language, OWL. The two main objectives for this formalization was to be consistent with the CMMI-DEV model 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. The formalization is presented in a number of increments—from more general concepts to more specific. A justification for the selection of the concepts and relations is given. To assess the validity of the formalization, a number of test cases for the scenario of automatic determination of the maturity level were developed. Generic OWL reasoners were then used to derive the maturity levels. While the test results were all positive, the real value of this formalization comes from the fact that it faithfully captured the main aspects of CMMI-DEV, a well established and accepted model of the assessment of the maturity of development processes, and that a generic inference engine was able to support the appraisal of the process maturity of an organization.

Keywords: CMMI-DEV; Process Maturity; Ontology; Automatic Inference; OWL

1. Introduction

The Capability Maturity Model Integration (CMMI) plays various roles [1]: a process improvement approach [2]; a way to describe the characteristics of effective processes [3]; a collection of the essential elements of effective processes for one or more bodies of knowledge; a process capability maturity model which aids in the definition and understanding of an organization's processes [4]; and more. In some texts it is referred to as "the CMMI model", while some others use "a CMMI model". This diversity of interpretations can be viewed as a manifestation of the dual role that this concept plays: a conceptual framework for describing characteristics of business processes vs. a characterization of the business process of a specific organization. In the first meaning, the CMMI is a collection of concepts. In the second, it is the concepts mapped to a specific enterprise. This kind of duality is not a unique feature of the CMMI modeling framework but is rather typical of many modeling frameworks. For instance, a UML model of a software system captures the concepts—the classes and the associations among the classes. Then a specific run time system is an instantiation of such constants. In the Semantic Web domain the concepts of a domain are called Ontology, while the instances of the concepts are called Annotations (also sometimes referred to as a markup [5]).

The CMMI Product Suite includes various components and a CMMI Framework, which is used to generate multiple models and related training and appraisal materials. The components used to generate a specific model are called a constellation. The models are categorized by representations and the types of processes. There are two representations of the model: continuous and staged. The continuous representation enables selections on the order of improvement with respect to an organization's business objectives, and allows comparisons within and between organizations by process areas. The staged representation presents a sequence of improvements, advancing through a predefined and proven path of successive levels, where each level serves as a basis for the next maturity level. It allows comparisons within and between organizations by maturity levels. It provides a single rating for the appraisal results. In this paper we focus on the staged representation. The newer releases of the CMMI models include CMMI for Acquisition [6], CMMI for Development [1,7] and CMMI for Services [8]. In this

paper we deal with CMMI-DEV [7], *i.e.*, the specialization of the CMMI to development. CMMI-DEV is used by various software organizations to assess the maturity of their development processes and to plan improvements.

The staged CMMI-DEV model distinguishes five maturity levels as shown in **Figure 1**. According to CMMI-DEV, the highest achievable objective for an organization is to be at the maturity level 5.

In order to assert a specific maturity level, an organization must follow an appraisal process which needs to show that the organization satisfies various requirements specified in the CMMI-DEV model. This model is rather complex since it includes many concepts that are interrelated in quite complicated ways. In order to assess the maturity level, an organization may proceed in two steps. First, it can use experts who are familiar with the CMMI-DEV model; they can verify that specific practices are implemented. In the second step, the relations among practices, goals and process areas need to be checked. Checking all the relationships among the concepts is a very tedious process.

The matter is complicated even more when some elements in the model change. For instance, a new type of practice is accepted by the industry or a new goal is identified as necessary to satisfy a specific process area. And one more issue is that people are prone to errors. In other words, the tedious process of verifying the relationships between various practices, goals and process areas in an organization may be unintentionally erroneous.

A computer based support tool would be able to alleviate some of the problems mentioned above. A computer tool would not be prone to errors. It could be faster. It would be cheaper to use than people. Moreover, if designed properly, any modification in the CMMI-DEV model could be relatively easily implemented, and a new version of the tool could be made available to the users in a relatively short time.

The above discussion provides a motivation for the work presented in this paper. In order to address the issues mentioned above, a computer-interpretable version of the CMMI-DEV model would have to be developed. By "computer-interpretable", we mean a version of the model that could be used by a computer (an inference engine) to actually infer whether a given organization has achieved a specific maturity level, or infer the highest level that it can be classified at. For this task, the computer would have to be provided with appropriately structured input about the software engineering process areas, goals and practices in the given organization. In order to facilitate such an inference task, a representation of the CMMI-DEV model would need to have computer executable semantics.

Towards these goals, in this paper we provide an attempt at a faithful, although not complete, representation of the CMMI-DEV model in the Web Ontology Language (OWL) [9], a language with formal, computer-executable semantics. The first version of such a representation was developed for CMMI-SW and described in [10]. To avoid

Level	Focus	Process Areas	
5 Optimizing	Continuous Process Improvement	Organizational Performance Management (OPM) Causal Analysis and Resolution (CAR)	Quality Productivity
4 Quantitatively Managed	Quantitative Management	Organizational Process Performance (OPP) Quantitative Project Management (QPM)	
3 Defined	Process Standardization	Requirements Development (RD) Technical Solution (TS) Product Integration (PI) Verification (VER) Validation (VAL) Organizational Process Focus (OPF) Organizational Process Definition (OPD) Organizational Training (OT) Integrated Project Management (IPM) Risk Management (RSKM) Decision Analysis and Resolution (DAR)	
2 Managed	Basic Project Management	Requirements Management (REQM) Project Planning (PP) Project Monitoring and Control (PMC) Supplier Agreement Management (SAM) Measurement and Analysis (MA) Process and Product Quality Assurance (PPQA) Configuration Management (CM)	
1 Initial			Risk Rework

Figure 1. CMMI-DEV Model.

confusion, the term "faithful" needs to be clarified. In this paper, faithful means that a generic OWL inference engine can correctly derive the maturity level of an organization's development process, provided the engine is supplied with the data about the organization as described in this paper.

A computer-interpretable version of the CMMI-DEV model could also play the role of an enabler of the interoperability among software process management systems. When two process management systems share a formalization of the same model, they could exchange information about the process areas covered by particular processes, their goals and practices that are in use. The important aspect of this scenario is that the two systems would be able to "understand" the meaning of the exchanged information, in the sense that they would be able to (automatically) draw conclusions of the implications of a specific process areas, goals and practices.

This paper is organized as follows. In the next section we discuss the main usage scenario that we considered as a potential application of our CMMI-DEV model formalization, *i.e.*, the automatic inference of the maturity levels. Section 3 provides a description of the CMMI-DEV model formalization. The formalization is introduced in incremental fashion, in three increments, from the most general classes and properties to the lower-level subclasses. Section 4 presents a description of the approach to the validation of the formalization. In particular, it describes the test data and the tools used for inferring maturity levels. Section 5 gives a brief description of related work. And finally, Section 6 presents our conclusions and suggestions for future research.

2. Usage Scenario of the CMMI-DEV Formalization

In the intended usage scenario, an organization collects information about its specific and generic practices that it uses in its software development process, expresses this information in terms of the CMMI-DEV formalization and then invokes a model interpreter tool to check the consistency of the representation and to derive the levels of maturity of the organization's development processes. While it is possible that some of the practices in an organization have different names than the practices listed in the CMMI-DEV model, it would be the responsibility of the organization to associate its local practices with the practices recognized in the model. If the model is logically inconsistent, some remedial action would have to be taken to eliminate the source(s) of inconsistency.

As was mentioned earlier, in this paper we describe a formalization of the CMMI-DEV model in the Web Ontology Language (OWL) [9], a primary language for the Semantic Web [11]. According to the approach practiced in the Semantic Web, the modeling consists of two phases: 1) The representation of the generic concepts of a domain as an ontology that includes classes, properties (relations) and constraints; and 2) The capturing of the instances of the classes and the properties that are specific to a case being modeled by the ontology.

Since we used OWL as the language to formalize CMMI-DEV model, we followed the same approach as in the Semantic Web. First, we formalized CMMI-DEV as an ontology. This ontology captures the main concepts of this model. Note that the use of the term "ontology" makes use of the interpretation of this notion that is used in knowledge representation [12] and not as it is used in philosophy [13]. We call this formalization the CMMI-DEV Ontology. This ontology is then used to annotate specific and generic practices of a specific organization. In the next step, a generic OWL reasoner, e.g., Pellet [14], Racer [15], BaseVISor [16] or OWLIM 2.9.1 [17], is used to check the consistency of the representation and then to derive the classification of the level of maturity of the organization's development process.

3. Structure of the CMMI-DEV Ontology

The CMMI-DEV model is a very complex structure. Its description is provided in natural language text plus some graphics. In some cases, it is supported by figures and tables. A graphical representation of the CMMI model (as presented in [7]) is shown in **Figure 2**. This graphical representation, along with textual descriptions, was helpful in selecting concepts to be captured in the CMMI-DEV formalization.

The OWL-DL code for the full-scale ontology is located at:
http://www.ece.neu.edu/groups/scs/onto/CMMI-DEV/.
The ontology includes 313 classes and three kinds of properties. Some of the classes are primitive (or declared), *i.e.*, they have some necessary restrictions that need to be satisfied by an individual to be an instance of one of those classes. The defined classes are those that have both necessary and sufficient conditions for an instance to be member of such a class. 86 classes in the CMMI-DEV ontology are defined classes. For these classes, not only an individual needs to satisfy the restrictions of the class definition to be an instance of a given class, but also an OWL reasoner can infer whether a given individual is a member of such a class. This fact is important for the automatic inference of the membership of an organization in particular maturity levels of the CMMI-DEV model.

3.1. Top Level Classes and Properties

The goal of the work described in this paper was to capture the main parts of the model, *i.e.*, identify various

concepts and relationships to be represented in the ontology. The top level of the ontology is shown in **Figure 3**. Concepts are represented as OWL classes and relationships as OWL properties.

Our first decision was to proceed in an incremental top-down fashion. In the first step (increment) we have identified four top-level classes: Maturity_Level, Process_Area, Goal and Practice. All these classes are shown in **Figure 3** as rectangles.

The decision to consider these four classes in the ontology was based on the statements in the CMMI document [7]. Maturity levels are the basis of classification in the staged CMMI-DEV model, so the Maturity_Level class had to be included. Goals are a required component of the CMMI-DEV model. Practices are expected components of the model. Process Areas are used in the CMMI-DEV document [7] as containers of both goals and practices. This is also indicated in **Figure 2**.

Relations among specific classes are shown in **Figure 3** as arrows with associated labels representing relation names. The top level of the ontology includes three relations, or properties in the terminology of OWL: consistsOf, satisfiedBy and achievedBy. We tried to choose relations names so that they somewhat reflect their nature within the model. The relationships among the four top-level classes are patterned upon the hierarchy shown in

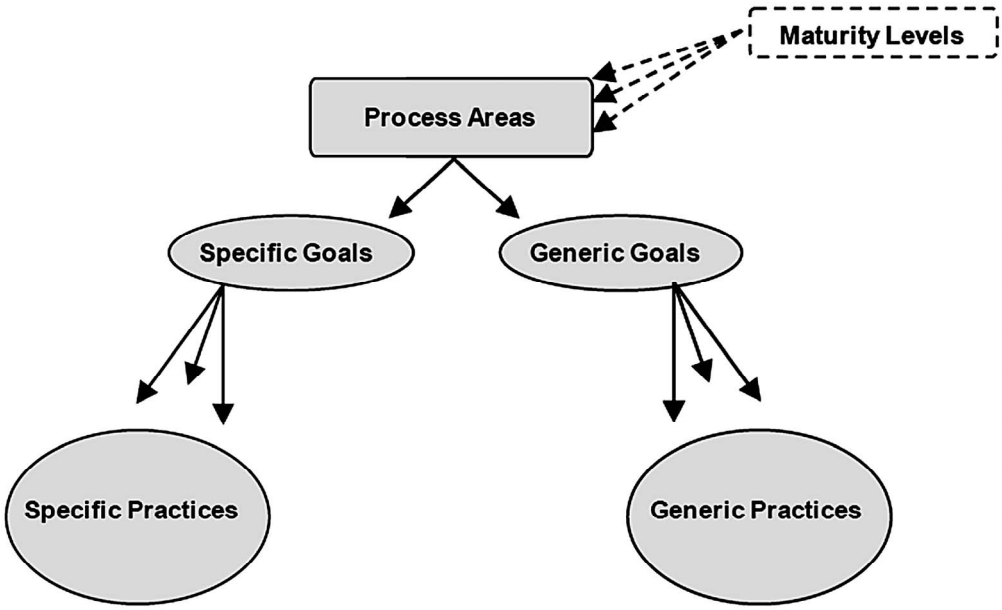

Figure 2. CMMI model: graphical representation.

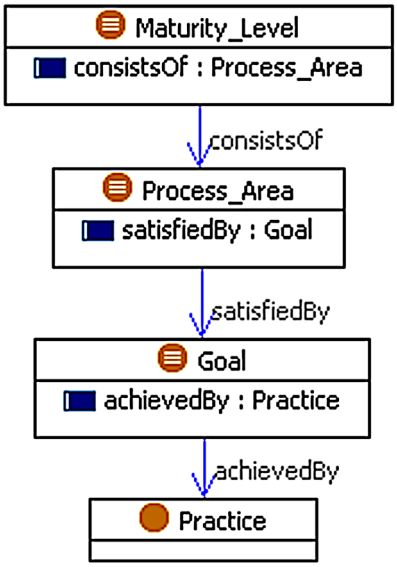

Figure 3. Top level of the CMMI-DEV ontology.

Figure 1 which suggests this kind of relationships.

The consistsOf property is supported by the CMMI-DEV description [7], which states that "*A maturity level consists of related specific and generic practices for a predefined set of process areas that improve the organizations overall performance*." Moreover, the CMMI-DEV document [7] states that: "*The maturity levels are measured by the achievement of the specific and generic goals associated with each predefined set of process areas*."

The existence of a satisfiedBy relationship between Process_Area and Goal is supported by the definition of process area in [7]: "*A cluster of related practices in an area that, when implemented collectively, satisfies a set of goals considered important for making improvement in that area*." We chose satisfiedBy rather than satisfy primarily because generic goal is defined in [7] as: "A required model component that describes characteristics that must be present to institutionalize processes that implement a process area." Another potential candidate for

the name of this relationship could be present.

Finally, the relationship between Goal and Practice was named achievedBy, mainly because specific practice is defined in [7] as: "An expected model component that is considered important in achieving the associated specific goal. The specific practices describe the activities expected to result in achievement of the specific goals of a process area."

Another possibility would be to introduce a ternary relation between Process_Area, Goal and Practice. The following statement in [7] seems to suggest that there is a linear chain of relations between Maturity_Level, Process_Area and Goal. "To reach a particular level, an organization must satisfy all of the goals of the process area or set of process areas that are targeted for improvement, regardless of whether it is a capability or a maturity level." Moreover, the analysis of tables in [7] that group practices shows that in fact only relationships between Process_Area and Goal, as well as between Goal and Practice are actually used. This revealed that only two binary relations can be used to represent the description of what might seem like a ternary relation.

The CMMI-DEV model describes relationships among the practices. This aspect was not modeled in our ontology, primarily because our focus at this time was on the ability to infer the maturity levels of an organization's development processes. The ontology would need to be expanded to capture such inter-practice relationships.

3.2 Second Increment of the CMMI-DEV Ontology

The second increment of the ontology introduces the classes at one level deeper in the ontology (**Figure 4**). Four new classes (two subclasses of Goal and two subclasses of Practice) and one new property are added to the classes shown in **Figure 3**. The subClassOf relation between two classes is represented in this figure by an arrow with a hollow arrow end pointing to the superclass.

The class Goal introduced in Section 3.1 does not have a direct corresponding component in the CMMI-DEV representation in **Figure 2**. Instead, **Figure 2** shows model components of "specific goals" and "generic goals". Nevertheless, we introduced Goal as a top-level class (**Figure 3**) and then added two subclasses, *i.e.*, Specific_Goal and Generic_Goal, in **Figure 4**. Similarly, the class Practice is introduced as a superclass of Specific_Practice and Generic_Practice. The primary reason for the introduction of these superclasses is to show the commonalities between generic and specific practices, and between generic and specific goals. They also play similar roles in the CMMI-DEV model. And finally, in this way we do not need to introduce additional properties between generic goals and generic practices, as well as between specific goals and specific practices. The achievedBy property is sufficient to represent both of the relations.

3.2.1. Maturity Levels

One of the primary considerations behind this work was the automatic inference of the maturity level from the data about generic and specific practices within a company. As with any formalization, the choice of the formalization language imposes some constraints on what can be represented in the ontology, as well as how it can be done. Since we chose OWL as the formalization language for our ontology we had to construct the ontology in such a way that the automatic inference of maturity levels from information about specific practices is possible.

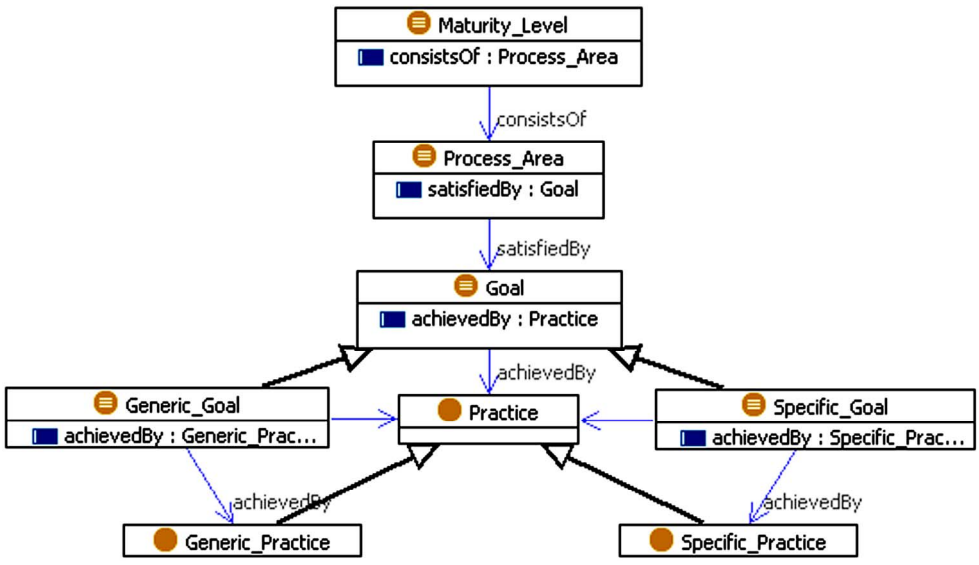

Figure 4. Second level of detail of the CMMI-DEV ontology.

OWL facilitates various kinds of inferences, e.g., subsumption, satisfiability, instance retrieval and type inference. Subsumption reasoning allows the inference that one class is a subclass of another. This inference is based upon the intentional definitions of the classes using primarily property restrictions—defining a class as those individuals that on a given property have values from another class. Satisfiability reasoning allows one to infer whether a proposed type of individual (class) is satisfiable, *i.e.*, whether it can be instantiated concretely. Instance retrieval allows one to infer which of the individuals are instances of a particular class. Type inference derives the classes that a given individual is an instance of.

One of the first decisions that we had to make was how the concept of maturity level should be represented. In OWL, each maturity level could be modeled as a property, an instance or a class. In the first case, there would be five properties corresponding to the five maturity levels. This option was rejected mainly because OWL does not have much support for defining properties in terms of other concepts and does not provide means for property inference. The second option would result in having one class—Maturity Level—with five individuals, each being a maturity level. This option was rejected for at least two reasons. First, according to [7], maturity levels are organized in a hierarchy where the higher levels include all of the features of the levels below them and introduce some additional features. This aspect can be captured by the OWL subclass relation and by the inheritance property associated with this relation. But it would be difficult, if possible at all, to capture this kind of relationship between two consecutive maturity levels if they were represented simply as instances of only one class. An additional problem with this conceptualization would be that this would imply that there should be only one way of achieving a given maturity level, which again would go against the spirit of CMMI. Consequently, we introduced the Maturity_Level class, as shown in **Figure 3**, and then modeled the dependencies among the maturity levels using the subClassOf property of OWL as shown in **Figure 5**.

In this conceptualization, a company can be identified with a maturity level. Since Maturity_Level is the lowest possible level and all of the features of this level are also included in all upper levels, a company can be safely identified with this level. This class thus plays a double role—as a generic class of maturity levels and as a class representing maturity level 1. Then an OWL reasoner can be used to infer whether the company also satisfies higher maturity levels. Thus the kind of inference used for this purpose is type inference, as described above. For such

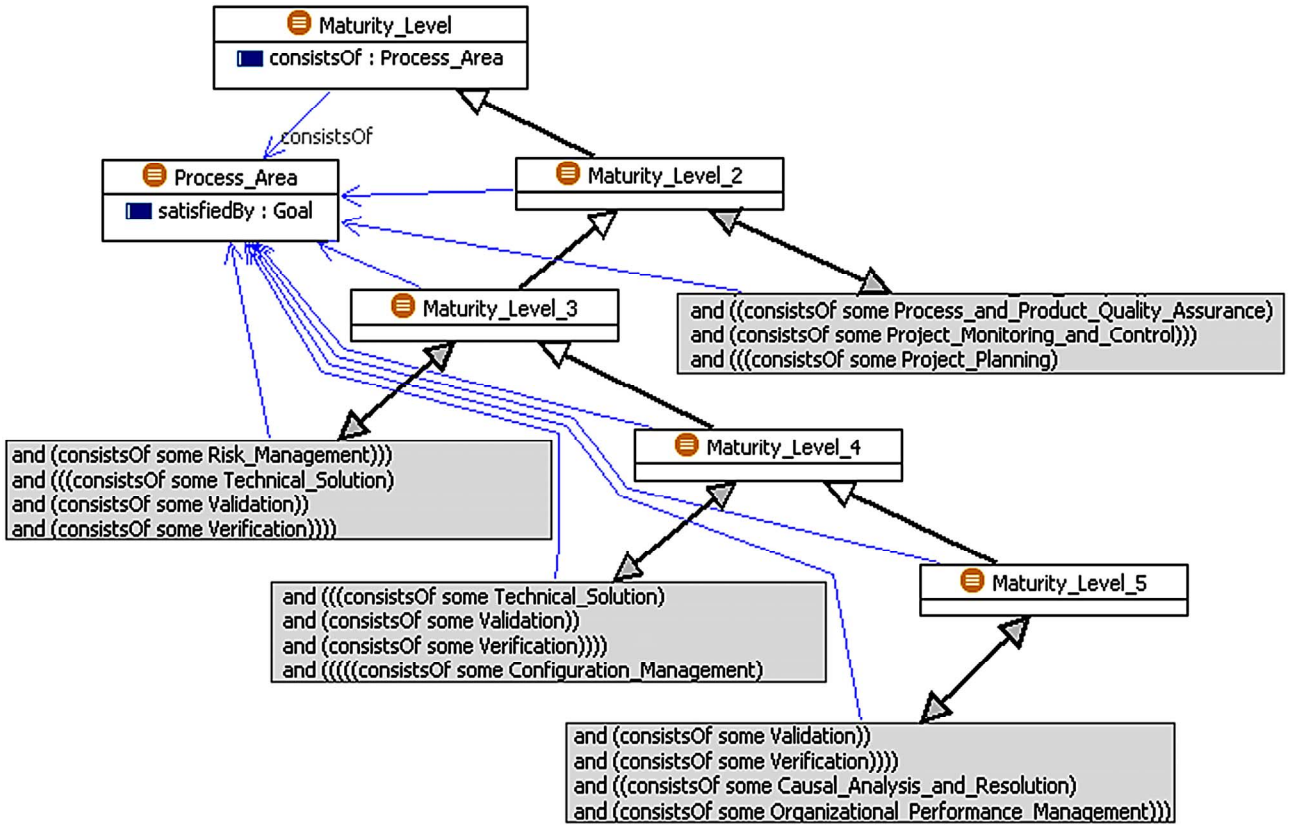

Figure 5. Maturity levels.

an inference to be possible, the classes representing particular maturity levels (except level 1) must be defined classes. Towards this aim, we defined appropriate restrictions for each maturity level class. In OWL, such restrictions are anonymous classes. **Figure 5** shows for each maturity level an anonymous class that is equivalent to a given maturity level. This is indicated by bidirectional "isa" arrows between a maturity level class and an anonymous class. The anonymous classes are defined by restriction expressions. Below we show how restrictions are represented in OWL. Due to the size of the OWL code, only a small part of the restriction on the Maturity_Level_2 class is shown.

```
<owl:intersectionOf rdf:parseType="Collection">
<owl:Restriction>
<owl:someValuesFrom
rdf:resource="#Configuration_ Management"/>
<owl:onProperty>
<owl:ObjectProperty rdf:about="#consistsOf"/>
</owl:onProperty>
</owl:Restriction>
<owl:Restriction>
<owl:onProperty>
<owl:ObjectProperty rdf:about="#consistsOf"/>
</owl:onProperty>
<owl:someValuesFrom rdf:resource="#Measurement_
and_Analysis"/>
.....
</owl:Restriction>
</owl:intersectionOf>
```

Since the Maturity_Level_2 class has a number of restrictions, they are captured as an intersection of particular restrictions. In this case we show that each instance of Maturity_Level_2 must have at least one association with Configuration_Management and one with Measurement_and_Analysis through the consistsOf property. In plain English terms, this means that a company to be classified as an instance of maturity level 2 must include in its software process the process areas of Measurement and Analysis and Configuration Management. All restrictions can be viewed at http://www.ece.neu.edu/groups/scs/onto/CMMI-DEV/cmmi.owl.

3.3. Third Increment of the CMMI-DEV Ontology

In this increment, the ontology of the CMMI-DEV model is expanded by providing definitions for the subclasses of Process_Area, Specific_Goal, Generic_Goal, Specific_Practice and Generic_Practice. Due to the relatively large size of the ontology, it is difficult to show it in a graphical form. For this reason, in Table 1 we show the definition of one subclass—Requirements_Management.

A full version of this table would show all the subclasses of Process_Area in the first column. We chose the names of the subclasses of Process_Area that are similar to the process areas in the CMMI-DEV model. They are grouped under four subclasses, one for each maturity level: Process_Area_Level_2, Process_Area_Level_3, Process_Area_Level_4 and Process_Area_Level_5. Requirements_Management shown in Table 1 is a subclass of Process_Area_Level_2.

The second column contains the Specific_Goal and Generic_Goal subclasses. Following the convention used in the description of the CMMI-DEV model [7], the name of each subclass is prefixed by SG for Specific_Goal and GG for Generic_Goal. Following the prefix of SG, there is a number, which acts as an enumerator for the subclasses of Specific_Goal and Generic_Goal. So SG_1_Manage_Requirements in Table 1 is the class for the specific goal numbered 1. The convention for the GG prefix accepted in [7] is that the number after the GG prefix is an indicator that shows the corresponding Process_Area_Level_<x>, where x is an integer. So the class GG_2_Institutionalize_a_Managed_Process represents a goal associated with the process areas associated with the maturity level 2.

The third column contains the Specific_Practice and Generic_Practice subclasses. The name of each subclass is prefixed by SP for Specific_Practice and GP for Generic_Practice. Following the prefix, there are two numbers separated by a dot, where the first number corresponds to a Specific_Goal or a Generic_Goal. The second number enumerates the subclasses of Specific_Practice and Generic_Practice.

The subclasses at a row are related to each other through properties and restrictions. The subclasses in column one are restricted to such instances that on property satisfiedBy have at least one (existential restriction) value from a specific subclass of Goal. In other words, the subclasses of Process_Area are defined by the satisfiedBy property and a subclass of Specific_Goal or Generic_Goal. The Goal class is defined by the allValuesFrom restriction (necessary and sufficient) on property achieved By with values in the class Practice. However, a relatively rich subclassification of Goal provides more information than this restriction. Each subclass of Goal is defined as an existential restriction to particular subclasses of Practice on property achievedBy. So for instance, SG_1_ManageRequirements must have at least one of the five practices as value of achievedBy.

In total, there are 49 Specific_Goal subclasses. In order to enhance the readability of the ontology, the 49 subclasses are grouped into 22 intermediate subclasses, which have the naming convention <name>_Goal, where name is the name of the Process_Area that is related to

the subclass of Specific_Goal grouped under this subclass. For instance, the goal SG_1_Manage_Requirements falls under the Requirements_Management_Goal intermediate class. These intermediate subclasses of Specific_Goal, which are used solely for the grouping, are not shown in Table 1.

Table 1. Example relationships among subclasses of Process_Area, Goal and Practice.

Process Area	Goal	Practice
Requirements Management	SG 1 Manage Requirements	SP 1.1 Understanding Requirements
		SP 1.2 Obtain Commitment to Requirements
		SP 1.3 Manage Requirements Changes
		SP 1.4 Maintain Bidirectional Traceability of Requirements
		SP 1.5 Ensure Alignment between Project Work and Requirements
	GG 2 Institutionalize a Managed Process	GP 2.1 Establish an Organizational Policy
		GP 2.2 Plan the Process
		GP 2.3 Provide Resources
		GP 2.4 Assign Responsibility
		GP 2.5 Train People
		GP 2.6 Control Work Products
		GP 2.7 Identify and Involve Relevant Stakeholders
		GP 2.8 Monitor and Control the Process
		GP 2.9 Objectively Evaluate Adherence
		GP 2.10 Review Status with Higher Level Management

4. Validation of the Formalization

In the previous sections we showed how the CMMI-DEV ontology presented in this paper was constructed. The main purpose of this discussion was to show the relation between the CMMI-DEV model described in [7] and the ontology, and show that the ontology is a relatively faithful formalization of the model. The goal was to convince the reader that this is actually the case. Obviously, this is a subjective judgment. Since, as stated in the Introduction section, the main usage scenario for this ontology that guided its development was the automatic inference of the maturity level of an organization's development process, we also tested this formalization on a number of cases. For this purpose, a set of test cases was developed and then OWL inference engines were used for the automatic inference of facts entailed by the ontology. In particular, the derivation of the maturity level of an organization was demonstrated. In this section we describe some of our experiments.

4.1 Test Data

We used the results of the SEI Appraisal Program [18] to assess the validity of our formalization. SEI has designed "the Standard CMMI Appraisal Method for Process Improvement (SCAMPI^{SM}) to provide benchmark quality ratings relative to CMMI models" [19]. These results show the assigned maturity level of the staged version of CMMI-DEV model for the appraised organizations, based on the process areas determined by the appraisal method. In order to attain a maturity level, an organization should have "satisfied" or "not applicable" ratings for the process areas that maturity level consists of. For our experiments, fifteen organizations were selected from the appraisal results. The appraisal results for the fifteen organizations and the ratings for the process areas applied by each organization are shown in **Table 2**. The organizations in **Table 2** are labeled O1 through O15. Each row shows which organizations cover the process area represented by the row and whether the process area is satisfied (S), not applicable (NA) or out of scope (OS). **Table 3** shows the legend for all the labels in **Table 2**.

All the data from **Table 2** was annotated in terms of the CMMI-DEV ontology so that it could be processable by an OWL reasoner. Moreover, instances of all the classes from the CMMI-DEV ontology (Generic_Goal, Specific _Goal, Generic_Practice and Specific_Practice) had to be created. Those instances need to be present for a particular organization in order to satisfy the restrictions of the maturity level that the organization has been assigned. The overall number of instances that had to be created was quite large. For instance, for an organization to be at the maturity level 5, at least 255 instances need to be created. For other levels these numbers are 215 for level 3 and 92 for level 2. All fifteen annotation files, one for each company, can be viewed at http://www.ece.neu.edu/groups/scs/onto/CMMI-DEV/.

4.2. Testing

As the first step, both the CMMI-DEV ontology and the OWL files that contained the test cases were checked for consistency using an ontology consistency checker ConsVISor [20]. The result of the tests on the final version of these files was that all the ontologies were consistent.

In the next step, two OWL reasoners were used to derive development process maturity levels of the 15 organizations. The reasoners were BaseVISor [16], OWLIM 2.9.1 [17] and Pellet 2.3.0 [14]. All three inference engines are available for free for research purposes. All of them were able to process the reasoning tasks for all 15

Table 2. Test cases.

	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12	O13	O14	O15
AppR	3	5	3	5	5	3	2	5	2	3	2	5	3	2	5
REQM	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
PP	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
PMC	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
SAM	NA	S	S	NA	NA	S	S	NA	NA	S	S	S	S	NA	NA
MA	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
PPQA	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
CM	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
RD	S	S	S	S	S	S	OS	S	OS	S	OS	S	S	OS	S
TS	S	S	S	S	S	S	OS	S	OS	S	OS	S	S	OS	S
PI	S	S	S	S	S	S	OS	S	OS	S	OS	S	S	OS	S
VER	S	S	S	S	S	S	OS	S	OS	S	OS	S	S	OS	S
VAL	S	S	S	S	S	S	OS	S	OS	S	OS	S	S	OS	S
OPF	S	S	S	S	S	S	OS	S	OS	S	OS	S	S	OS	S
OPD	S	S	S	S	S	S	OS	S	OS	S	OS	S	S	OS	S
OT	S	S	S	S	S	S	OS	S	OS	S	OS	S	S	OS	S
IPM	S	S	S	S	S	S	OS	S	OS	S	OS	S	S	OS	S
RSKM	S	S	S	S	S	S	OS	S	OS	S	OS	S	S	OS	S
DAR	S	S	S	S	S	S	OS	S	OS	S	OS	S	S	OS	S
OPP	OS	S	OS	S	S	OS	OS	S	OS	OS	OS	S	OS	OS	S
QPM	OS	S	OS	S	S	OS	OS	S	OS	OS	OS	S	OS	OS	S
OPM	OS	S	OS	S	S	OS	OS	S	OS	OS	OS	S	OS	OS	S
CAR	OS	S	OS	S	S	OS	OS	S	OS	OS	OS	S	OS	OS	S

Table 3. Legend for the test cases from Table 2.

Acronym	Description	Acronym	Description
AppR	Appraisal Result	OPF	Organizational Process Focus
REQM	Requirements Management	OPD	Organizational Process Def.
PP	Project Planning	OT	Organizational Training
PMC	Project Monitoring and Cntrl	IPM	Integrated Project Mngmnt
SAM	Supplier Agreement Mngmnt	RSKM	Risk Management
MA	Measurement and Analysis	DAR	Decision Analysis & Resol.
PPQA	Process & Prod. Qual. Assrn.	OPP	Organizational Proc. Perf.
CM	Configuration Management	QPM	Quality Project Mngmnt
RD	Requirements Development	OPM	Org. Performance Mngmnt
TS	Technical Solution	CAR	Causal Analysis & Resol.
PI	Product Integration	S	Satisfied
VER	Verification	NA	Not Applicable
VAL	Validation	OS	Out of Scope

cases within seconds. All the test cases resulted in correct (expected) inference. In other words, for all the fifteen test cases listed in **Table 2**, BaseVISor, OWLIM and Pellet derived that the organizations satisfied the levels specified in the table, as well as all the levels below the highest level.

5. Related Work

Our literature search has identified a number of efforts on implementing formal ontologies for software engineering in general ([21]). However, our interest was specifically related to the use of automatic inference of the CMMI-DEV maturity levels of software engineering processes. Below we list the efforts that are related to the use of ontologies for software engineering process improvement in general and the CMMI-DEV model in particular.

The work reported in this paper is based on, and is an extension of, the (unpublished) work reported at the 2006 ISWC Workshop on Semantic Web Enabled Software Engineering (SWESE) [22]. The ontology described in [22] was modified to account for the migration of the CMMI-SW [23] to CMMI-DEV V.1.3 [7].

Mendes and Abran [24] have initiated a project for the development of a software engineering domain ontology based on the Software Engineering Body of Knowledge (SWEBoK) [25]. Although OWL was used to implement the ontology, we were not able to access the ontology and possibly base our ontology on some results of this work. Moreover, the goal of the development of the SWEBoK ontology was different than ours. As stated in [24], their project had five objectives: "characterizing the software engineering discipline contents; providing access in terms of topics to the software engineering body of knowledge; promoting a consistent view of software engineering; clarifying the location and setting the boundaries of software engineering with respect to other related disciplines; creating a basis for curriculum development and individual certification."

Liao, Qu and Leung [26] introduced an ontology named Software Process Ontology (SPO) to express software processes at the conceptual level. The focus of this ontology was on the software engineering process and the integration of various process assessment paradigms. For this reason, the ontology has two subclasses for linking SPO with ontologies for CMMI and ISO/IEC 15504. However, to the best of our knowledge, this ontology did not support the inference of the maturity levels.

Chang-Shing Lee, *et al.* [27,28] conducted research on the application of ontologies to project management. As part of this research, they have reported on the development of a "CMMI ontology". Their ontology includes many concepts that come out of the CMMI model [7], like the key process areas of Requirements Management and Software Project Planning. However, since the goal of their project is different from ours, their ontology has very little resemblance with the CMMI-DEV ontology presented in this paper. The structure of their ontology is based on their own conceptualization of both the classes and the relations among the classes, while in our case the intent was to just capture the CMMI-DEV concepts (as faithfully as possible) in a formal representation rather than introduce a new conceptualization. Consequently, while the ontology discussed in [27,28] serves the purpose intended by its authors, it would not be possible to infer the maturity level of an organization by using off-the-shelf ontology inference engines.

Gazel, Sezer and Tarhan [29] created a software process assessment tool called Ontology-based CMMI Mapping and Querying Tool (OCMQT) which is an Eclipse plug-in extension to EPF Composer. A CMMI ontology based on CMMI-Dev 1.2 was created. Users can view a CMMI ontology, create a software process ontology, construct a mapping ontology which maps the software process ontology to the CMMI ontology and query all the three ontologies. It is stated in [29] that the ontology captures both continuous and staged representations of CMMI. However the details of the ontology are not provided in the paper. Although some similarities to the ontology described [22] seem to be apparent, e.g., the property names, it is not clear whether this ontology supports the inference about the maturity levels.

Rungratri and Usanavasin [30] have constructed an ontology called Project Assets Ontology (PAO) and a CMMI Maturity/Capability assessment tool called CMMI v.1.2 based Gap Analysis Assistant Framework (CMMI-GAAF). PAO is an extension to the older version of the CMMI ontology by Soydan and Kokar [22]. It includes typical work products, data type properties defining project assets and evidences of information related to the typical work products. CMMI-GAAF uses project assets from an organization and PAO to make an assessment on the CMMI practices and goals performed and achieved. The tool produces a report called Practice Implementation Indicator Description (PIID), which shows the achieved and absent practices and goals to achieve maturity levels in an organization. While this paper presents an ontology and a tool for comprehensive assessment of an organization in achieving maturity levels, it doesn't make use of ontology inferencing capabilities.

A number of efforts were devoted to developing conceptual ontologies (although not represented in a formal knowledge representation language). The reasoning components of these systems follow a specially developed algorithm that processes fuzzy rules. Wang, *et al.* [31] use a fuzzy ontology and ontology-based semantic inferencing to a CMMI-based system for student performance

in the After School Alternative Program in Taiwan. Also, Wang, *et al.*, Lee, *et al.*, [32] and Wang, *et al.* [33] have developed an ontology-based intelligent estimation agent which uses fuzzy inference and a CMMI-based project planning ontology. This agent estimates the total cost of a project. Lee, Wang, Liu and Lin [34] developed a Customer Relationship Management (CRM) ontology using CMMI project planning. The intent was to use the ontology in business process planning. Lee and Wang [35] worked on a project in which they used an ontology, natural language processing and the intelligent agent technology for generating summaries of evaluations of the software engineering process with respect to CMMI. In a similar vein, Lee, *et al.* [36] applied a CMMI based ontology approach to project monitoring and control. Sang Hun Lee, *et al.* [37] discussed the relation between the CMMI reference model and the OTK ontology development methodology.

The need for the use of ontologies in software engineering has been recognized by many other researchers and organizations. In most cases the purpose of developing a software engineering ontology has been to establish a common vocabulary and to provide formalization of software engineering concepts ([21,38-40]). To the best of our knowledge, none of the existing ontologies allow for the automatic inference of maturity levels as described in this paper.

6. Conclusions and Future Work

The main purpose of the work described in this paper was to demonstrate the capability of automatic classification of maturity levels based upon some characteristics of the software engineering processes used by an organization. Towards this aim, a comprehensive formalization of the CMMI-DEV model as an ontology was implemented in OWL-DL. The ontology includes 313 classes and three properties. 86 of the classes are defined classes; others are primitive classes.

The ontology has been validated on fifteen cases (fifteen organizations) extracted from the set of results developed by the SEI Appraisal Program. In order to annotate these organizations, a large number of instances had to be added to the base CMMI-DEV ontology. The number of instances ranged from 92 to 255, depending on the maturity level of an organization. All of the fifteen test cases are accessible at:
http://www.ece.neu.edu/groups/scs/onto/CMMI-DEV/.
The particular test cases at this URL are identified as cmmi_test1.owl through cmmi_test15.owl.

The BaseVISor, Pellet and OWLIM inference engines were used successfully to derive the maturity levels of the organizations based on the supplied data, and for all of the test cases the inference engines derived the same conclusions as in the appraisal results.

Since the CMMI-DEV Ontology has computer-executable semantics, it can be used for automatic reasoning about the maturity levels of organizations, based upon some data provided by the organization. An OWL reasoner can be used for this purpose. The ontology could also be used in other scenarios, including process improvement and process optimization. And finally, the ontology can be used for passing information about particular aspects of the software engineering processes, both within and among software organizations.

Although the validation results indicate that the ontology faithfully captures the concepts and constraints of the CMMI-DEV model, the ultimate value of the ontology can only be appreciated if the community accepts it and decides to use it for both capturing process information and for interchange of information among various tools and various users. Towards this aim, we have published the ontology on our web site. This ontology can be treated as a "core ontology" that can be extended to more fully capture the concepts that are needed by the potential users.

The ontology is available at:
http://www.ece.neu.edu/groups/scs/onto/CMMI-DEV/cmmi.owl.

REFERENCES

[1] M. B. Chrissis, M. Konrad and S. Shrum, "CMMI: Guidelines for Process Integration and Product Improvement," 3rd Edition, Addison-Wesley, Boston, 2011.

[2] Carnegie Mellon Software Engineering Institute, "What Is CMMI?" 2008. http://www.sei.cmu.edu/cmmi/general/.

[3] Carnegie Mellon Software Engineering Institute, "The CMMI Version 1.2 Overview presentation," 2008. http://www.sei.cmu.edu/cmmi/adoption/pdf/cmmi-overview07.pdf

[4] Capability Maturity Model, "Wikipedia," 2008. http://en.wikipedia.org/wiki/Capability_maturity_model

[5] J. Heflin, "OWL Web Ontology Language Use Cases and Requirements," 2004. http://www.w3.org/TR/webont-req/

[6] B. Gallagher, M. Phillipsand, K. Richter and S. Shrum, "CMMI-ACQ: Guidelines for Improving the Acquisition of Products and Services," 2nd Edition, Addison-Wesley, Boston, 2011.

[7] CMMI for Development, "CMMI-DEV V1.3," Technical Report, Software Engineering Institute, Pittsburgh, 2010.

[8] E. Forrester, B. Buteau and S. Shrum. "CMMI for Services: Guidelines for Superior Service," 2nd Edition, Addison-Wesley, Boston, 2011.

[9] W3C, "Web Ontology Language (OWL)," 2004. http://www.w3.org/2004/OWL/

[10] G. H. Soydan, "An OWL Ontology for Representing the

CMMI-SW Model," M.S. Thesis, Northeastern University, Boston, 2006.

[11] W3C, "Semantic Web Activity," 2006. http://www.w3.org/2001/sw/

[12] T. R. Gruber, "A Translation Approach to Portable Ontology Specifications," *Knowledge Acquisition*, Vol. 5, No. 2, 1993, pp. 199-220. doi:10.1006/knac.1993.1008

[13] M. Bunge, "Treatise on Basic Philosophy: Ontology I: The Furniture of the World," Springer, Dodrecht, 1977.

[14] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur and Y. Katz, "Pellet," 2005. http://pellet.owldl.com/

[15] V. Haarslev, R. Miller and M. Wessel, "Racer," 2005. http://www.sts.tu-harburg.de/ r.f.moeller/racer/

[16] C. J. Matheus, K. Baclawski and M. M. Kokar, "BaseVISor: A Triples-Based Inference Engine Outfitted to Process RuleML & R-Entailment Rules," *Proceedings of the Second International Conference on Rules and Rule Markup Languages for the Semantic Web*, Washington, November 10-11, 2006, pp. 67-74.

[17] A. Kiryakov, D. Ognyanov and D. Manov, "WISE Workshop," Springer, Dodrecht, 2005.

[18] Carnegie Mellon Software Engineering Institute, "List of Published SCAMPI Appraisal Results," 2005. http://seir.sei.cmu.edu/pars/pars_list_iframe.asp

[19] Carnegie Mellon Software Engineering Institute, "CMMI Appraisals," 2006. http://www.sei.cmu.edu/cmmi/appraisals

[20] K. Baclawski, M. M. Kokar, R. Waldinger and P. A. Kogut, "Consistency Checking of Semantic Web Ontologies," *Lecture Notes in Computer Science*, Vol. 2342, 2002, pp. 454-459.

[21] C. Calero, F. Ruiz and M. Piattini, "Ontologies for Software Engineering and Software Technology," Springer, Dodrecht, 2006.

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

[23] CMMI for Software Engineering, "Staged Representation, (CMMI-SW V1.1, Staged)," Technical Report, Software Engineering Institute, Pittsburgh, 2002.

[24] O. Mendes and A. Abran, "Software Engineering Ontology: A Development Methodology," *Metrics News*, Vol. 9, No. 1, 2004, pp. 68-76.

[25] P. Bourque and R. Dupris, "Guide to Software Engineering Body of Knowledge," IEEE Computer Society Press, Washington, 2005.

[26] L. Liao, Y. Qu and H. Leung, "A Software Process Ontology and Its Application," *ISWC2005 Workshop on Semantic Web Enabled Software Engineering*, 2005.

[27] C.-S. Lee, M.-H. Wang and J.-J. Chen, "Ontology-Based Intelligent Decision Support Agent for CMMI Project Monitoring and Control," *International Journal of Approximate Reasoning*, Vol. 48, No. 1, 2008, pp. 62-76.

[28] C.-S. Lee and M.-H. Wang, "Ontology-Based Computational Intelligent Multi-Agent and Its Application to CMMI Assessment," *Applied Intelligence*, Vol. 30, No. 3, 2007, pp. 203-219. doi:10.1007/s10489-007-0071-1

[29] A. T. S. Gazel and E. Sezer, "An Ontology Based Infrastructure to Support Ontology-Based Software Process Assessment," *Gazi University Journal of Science*, Vol. 25, No. 1, 2012, pp. 155-164.

[30] S. Rungratri and S. Usanavasin, "Project Assets Ontology (PAO) to Support Gap Analysis for Organization Process Improvement Based on CMMI v.1.2," *Proceedings of the Software Process*, Berlin, 10-11 May 2008, pp. 76-87.

[31] M.-H. Wang, Z.-R. Yan, C.-S. Lee, P.-H. Huang, Y.-L. Kuo, H.-M. Wang and B.-H. Lin, "Apply Fuzzy Ontology to CMMI-Based ASAP Assessment System," *IEEE World Congress on Computational Intelligence*, Barcelona, 18-23 July 2010.

[32] C.-S. Lee, M.-H. Wanga, Z.-R. Yana, C.-F. Lob, H.-H. Chuangb and Y.-C. Linb, "Intelligent Estimation Agent Based on CMMI Ontology for Project Planning," *Proceeding of the* 2008 *IEEE International Conference on Systems, Man and Cybernetics*, 12-15 October 2008, pp. 228-233.

[33] M.-H. Wang, C.-S. Lee, Z.-R. Yan, H.-H. Chuang, C.-F. Lo and Y.-C. Lin, "A Novel Fuzzy CMMI Ontology and Its Application to Project Estimation," *Journal of Internet Technology*, Vol. 9, No. 4, 2008, pp. 317-325.

[34] C.-S. Lee, Y.-C. Wang, W.-M. Liu and Y.-C. Lin, "CRM Ontology Based on CMMI Project Planning for Business Applications," *Proceedings of the 6th International Conference on Machine Learning and Cybernetics*, Hong Kong, 19-22 August 2007, pp. 2941-2946.

[35] C.-S. Lee and M.-H. Wang, "Ontology-Based Computational Intelligent Multi-Agent and Its Application to CMMI Assessment," *Applied Intelligence*, Vol. 30, No. 3, 2009, pp. 203-219. doi:10.1007/s10489-007-0071-1

[36] C.-S. Lee, M.-H. Wang, J.-J. Chen and C.-Y. Hsu, "Ontology-Based Intelligent Decision Support Agent for CMMI Project Monitoring and Control," *International Journal of Approximate Reasoning*, Vol. 48, No. 1, 2008, pp. 62-76. doi:10.1016/j.ijar.2007.06.007

[37] S. H. Lee, H.-J. Choi and S. Kim, "Analysis of Ontology Development Methodology Based on OTK and CMMI Level 4," *International Conference on Advanced Communication Technology*, 15-18 February 2009, pp. 585-590.

[38] R. A. Falbo and G. Bertollo, "Establishing a Common Vocabulary for Software Organizations Understand Software Processes," *EDOC International Workshop on Vocabularies, Ontologies and Rules for the Enterprise, VORTE'*, The Netherlands, 20 September 2005, pp. 25-32.

[39] R. A. Falbo, F. B. Ruy and R. D. Moro, "Using Ontologies to Add Semantics to a Software Engineering Environment," 17*th International Conference on Software Engineering and Knowledge Engineering*, Taiwan, 14-16 July 2005, pp. 151-156.

[40] T. Singarayan, "A Framework for Semantic-Enabled Software Engineering," 2007. www.semantic-conference.com/2007/conferenceglance.html

