An OWL Ontology for Representing the CMMI-SW Model

Gokhan Halit Soydan and Mieczyslaw M. Kokar

Department of Electrical and Computer Engineering
Northeastern University
Boston, Massachusetts, USA
{gsoydan,mkokar}@ece.neu.edu
http://www.ece.neu.edu/groups/scs/onto/CMMI/cmmi.owl

Abstract. This paper provides a short description of an ontology of the capability maturity model, CMMI-SW. The ontology is coded in a formal language, OWL. Some possible uses of the ontology are introduced. The ontology is presented in a number of increments – from more general concepts to more specific. Justification for the selection of concepts and relations is given. To assess the validity of the ontology, a number of test cases were used. An OWL reasoner was then used to derive the results. While the test results were all positive, the ontology has a high value only if the community accepts it. The main goal of this paper is to announce the existence of such an ontology and to invite the community to contribute to its further refinement.

1 Problem and Motivation

CMMI-SW is used by various software organizations to assess the maturity of their software development process. The results of the assessment are then used for process improvement.

The CMMI Product Suite uses a framework, which generates multiple models and related training and appraisal materials. The models are categorized by *representations* and *bodies of knowledge* that are included in a particular model. There are two representations of the model: *continuous* and *staged*. According to [1], 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. A 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. It allows comparisons within and between organizations by maturity levels. It provides a single rating for the appraisal results. We have focused on the staged representation.

There are four bodies of knowledge for CMMI models: systems engineering, software engineering, integrated product and process development and supplier sourcing [1]. In our work so far we have considered the CMMI model containing only software engineering, i.e., CMMI-SW.

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

Fig. 1. CMMI-SW Model

In order to achieve a specific level, an organization must satisfy various requirements specified in the CMMI-SW 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 usually hires experts who are familiar with the CMMI-SW model. Even for experts, 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. In such a case, the experts would need to be re-trained to be able to deal with the new version of the model.

And one more issue is that people are prone to errors. In other words, the tedious process of verifying the relationships between various practices and goals in an organization may be unintentionally mistaken.

It seems that a computer based support tool would be able to alleviate some of the problems mentioned above. A computer tool would not be prone to er-

rors. It could be faster. It would be cheaper to use. And companies could use such a tool without the need to hire external consultants. Moreover, if designed properly, any modification in the CMMI-SW 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-processable version of the CMMI-SW would have to be developed. By "computer-processable" we mean a version of the model that could be used by a computer to actually infer whether a given organization has achieved a specific maturity level, or infer the highest level that it can be classified at. In order to facilitate such a process, a representation of the CMMI-SW model would need to have computer executable semantics.

Towards this goal, in this paper we provide an attempt at a faithful, although not complete, representation of the CMMI-SW model in the Web Ontology Language (OWL) [2], a language with formal, computer-executable semantics. The main result of this work is an ontology (in OWL) that captures the main concepts of the CMMI-SW model. We call it the CMMI-SW Ontology.

2 Usage Scenario of the CMMI-SW Ontology

Since the CMMI-SW Ontology has computer-executable semantics, it can be used for automatic reasoning about the maturity levels of companies, based upon some data provided by the company. An OWL reasoner can be used for this purpose.

In the intended usage scenario, a company would insert information about its specific and generic practices that it uses in its software development process. While it is possible that some of the practices in an organization have different names than the practices listed in the CMMI-SW model, it would be the responsibility of the company to associate its local practices with the practices recognized in the model. To be more precise, the company's practices would have to be "annotated" in terms of the CMMI-SW Ontology, i.e., the company would have to use an editor to combine its description of the practices with the CMMI-SW Ontology.

The next step would be to invoke a reasoner to check the consistency of the ontology that now includes information specific to the company. If the ontology is inconsistent, some remedial action would have to be taken to eliminate the source(s) of inconsistency. Once the ontology is consistent, the user would have to invoke the reasoner to perform the "infer types" step. One of the results of this step would be the classification of the "instance" (i.e., the company) into one of the maturity levels of the CMMI-SW.

3 Structure of the CMMI-SW Ontology

The CMMI-SW model is a very complex structure. Its description is mainly natural language text. In some cases, it is supported by some tables. A graphical representation of the CMMI model (as presented in [1]) is shown in Figure 2. This graphical representation, along with textual descriptions, were helpful in selecting concepts to be captured in the CMMI-SW Ontology.

The OWL-DL code for the full-scale ontology is located at: `http://www.ece.neu.edu/groups/scs/onto/CMMI/cmmi.owl`. The ontology includes 309 classes and 4 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. 97 classes are *defined classes*, i.e., they have both necessary and sufficient restrictions associated with them. This means not only an individual needs to satisfy these restrictions to be a member, but also an OWL reasoner can infer that a given individual is a member of such a class. This fact is important since one of the reasons for developing this ontology was to be able to automatically infer the membership of an organization in particular maturity levels of the CMMI-SW model.

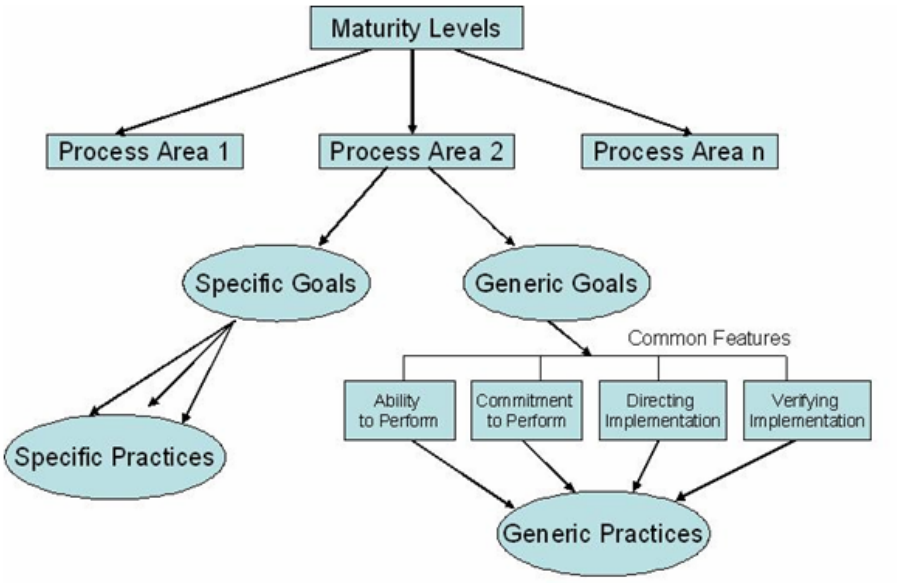

Fig. 2. CMMI Model: Graphical Representation

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.

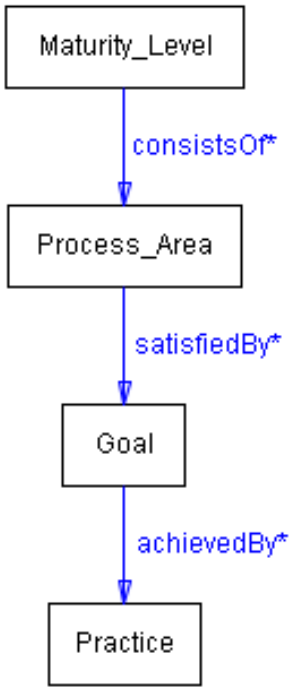

Fig. 3. Top Level of the CMMI Ontology

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`. One of the candidates for a top level class was `Common_Feature`, but we decided not to include it in the top level because it has relationships with `Generic_Practice`, i.e., with a subclass of `Practice`, rather than directly with `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 [1]. Maturity levels are the basis of classification in the staged CMMI-SW model, so the `Maturity_Level` class had to be included. Goals are a *required component* of the CMMI-SW model. Practices are *expected components* of the model. Although Process Area is not listed in [1] as a required component, the document makes it clear that processes must be implemented in order to address specific practices.

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`. The `consistsOf` property is a direct mapping from both Figure 1, which shows that particular process areas are associated with a maturity level, and the CMMI-SW description [1], which states that "Maturity levels consist of a predefined set of process areas." The relationships among the other three top-level classes were more complicated. Never in the model description, is it stated that a linear relationship (i.e., two binary relations) exists. The description seems to state the existence of a ternary relation. By analyzing the description of the model and putting the results into a table it became clear 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 seems like a ternary relation.

3.2 Second Increment of the CMMI-SW Ontology

The second increment of the ontology introduces the classes at one level deeper in the ontology (see Figure 4). Five new classes and one new property are built, where four classes are subclasses of the previously defined classes. The `subClassOf` relation between two classes is represented in this figure by an arrow with the "isa" label associated with the arrow.

The class `Goal` introduced in Section 3.1 does not have a direct corresponding component in the CMMI-SW 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 – `Specific_Goal` and `Generic_Goal` – in Figure 4. Similarly, the class `Practice` is made a superclass of `Specific_Practice` and `Generic_Practice`.

3.3 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. In OWL, this can be done either through class/subclass relations or through class definitions, which in turn are based on property restrictions. We could not use the former approach since the four top-level classes denote qualitatively different types of individuals. Therefore, we had to rely on the latter approach. Consequently, we modeled maturity levels using the `subclassOf` property of

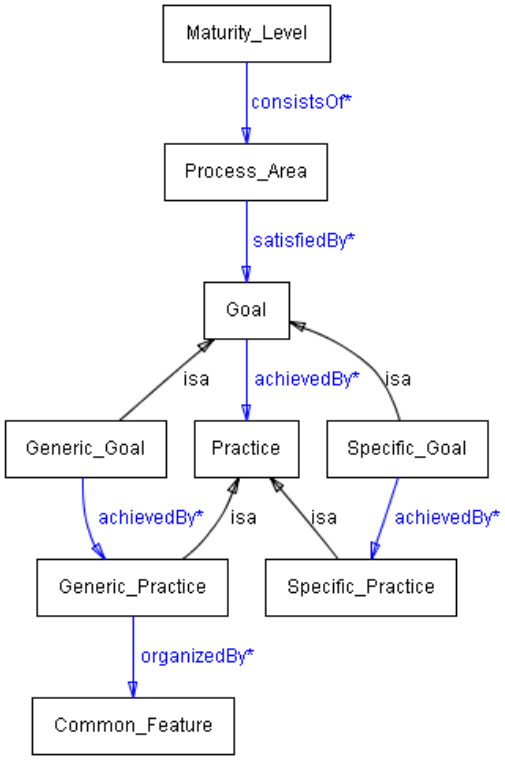

Fig. 4. Second Level of Detail of the CMMI Ontology

OWL as shown in Figure 5. This figure shows for each maturity level an anonymous class that is equivalent to a given maturity level. This is indicated by two "isa" arrows between a maturity level class and an anonymous class. The anonymous class is defined by restrictions. However, only the beginning part of the restriction expression is shown in the figure. All restrictions can be viewed at `http://www.ece.neu.edu/groups/scs/onto/CMMI/cmmi.owl`.

3.4 Third Increment of the CMMI-SW Ontology

In this increment, the ontology of the CMMI-SW model is completed by building definitions for `Maturity_Level` subclasses and for subclasses of `Process_Area`, `Specific_Goal`, `Generic_Goal`, `Specific_Practice`, `Generic_Practice` and `Common_Feature`.

The names of the subclasses of Process_Area are similar to the process areas in the CMMI-SW model. They are grouped under four subclasses: `Process_Area_Level_2`, `Process_Area_Level_3`, `Process_Area_Level_4` and `Process_Area_Level_5`.

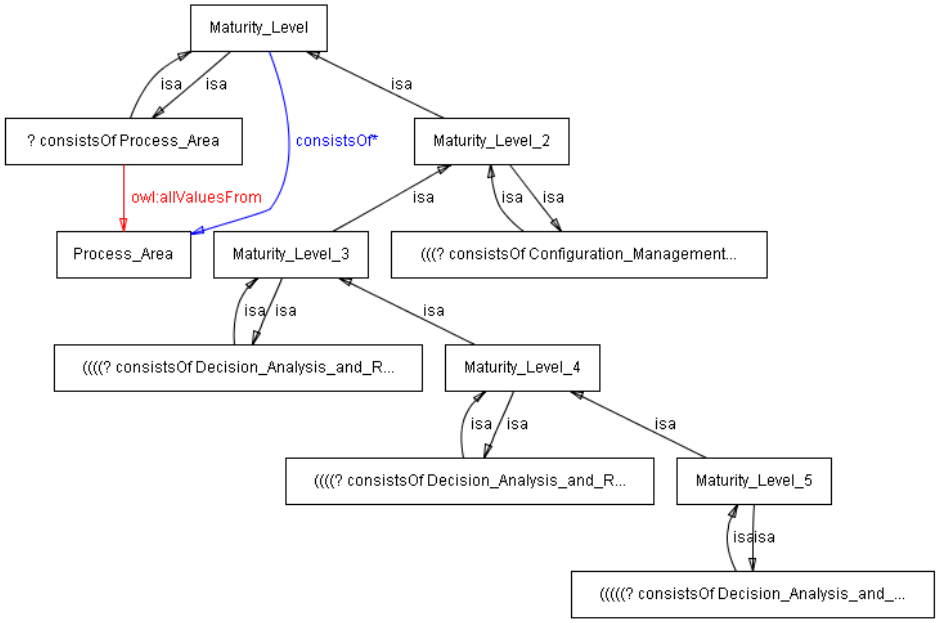

Fig. 5. Maturity Levels

Due to the complexity of the ontology, it is difficult to show it in a graphical form. For this reason, Table 1 is constructed to show the definitions of `Process_Area`, `Specific_Goal` and `Generic_Goal` subclasses. Table 1 shows only a small subset of the classes. The first column of the table contains the `Process_Area` subclasses. The second column contains the `Specific_Goal` and `Generic_Goal` subclasses. Following the convention used in the description of the CMMI-SW model [1], 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`. The convention for the `GG` prefix accepted in [1] is different than for `SG`, and thus, accordingly, the number after the `GG` prefix is an indicator that shows the corresponding `Process_Area_Level_<x>`, where `x` is an integer. 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 `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 spe-

cific 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`.

In total, there are 47 `Specific_Goal` subclasses. In order to enhance the readability of the ontology, the 47 subclasses are grouped into a number of 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 included in the table.

Table 1. Relationships among subclasses: An example

Process Area	**Goal**	**Practice**
Requirements Management	SG 1 Manage Requirements	SP 1.1 Obtain an Understanding of Requirements
	SG 1 Manage 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 Identify Inconsistencies 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 Manage Configurations
		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
	GG 3 Institutionalize a Defined Process	GP 3.1 Establish a Defined Process
		GP 3.2 Collect Improvement Information

The final component in the ontology is `Common_Feature`. In CMMI-SW model [1], the common features are defined as "Four common features organize the generic practices of each process area." In the definition, there are two com-

ponents, which are common features and generic practices. The definition relates common features to practices via the verb "organize". According to Figure 2 there should be a relationship from generic goals to common features and from common features to generic practices. However, there is no written definition in the CMMI-SW model, which shows a relationship between generic goals and common features. Since `Goal` is already defined in terms of `Practice`, and in order to maintain the domains and ranges as in Figure 2, the relationship from common features to generic practices is inverted. The property `organizedBy` is introduced with the domain `Generic_Practice` and range `Common_Feature`. `Generic_Practice` is then defined by a universal restriction on property `organizedBy` to the class `Common_Feature`.

Each subclass of `Generic_Practice` is defined by `organizedBy` property and a subclass of `Common_Feature`. To define practices, an existential restriction is asserted on the `organizedBy` property to an individual of a subclass of `Common_Feature`. In Table 2, this definition is represented by showing all the subclasses of `Generic_Practice` and the related subclasses of `Common_Feature` in the same row.

Table 2. Relationships between subclasses of `Generic_Practice` and `Common_Feature`

Generic Practice	**Common Feature**
GP 2.1 Establish an Organizational Policy	CO Commitment to Perform
GP 2.2 Plan the Process	AB Ability to Perform
GP 2.3 Provide Resources	AB Ability to Perform
GP 2.4 Assign Responsibility	AB Ability to Perform
GP 2.5 Train People	AB Ability to Perform
GP 2.6 Manage Configurations	DI Directing Implementation
GP 2.7 Identify and Involve Relevant Stakeholders	DI Directing Implementation
GP 2.8 Monitor and Control the Process	DI Directing Implementation
GP 2.9 Objectively Evaluate Adherence	VE Verifying Implementation
GP 2.10 Review Status with Higher Level Management	VE Verifying Implementation
GP 3.1 Establish a Defined Process	AB Ability to Perform
GP 3.2 Collect Improvement Information	DI Directing Implementation

4 Validation of the Ontology

To validate the ontology we developed a set of test cases. Then we attempted to use an OWL inference engine to use automatic inference to infer facts that are entailed by the ontology. In particular, we focussed on the derivation of the maturity level of an organization. In this section we describe some of our experiments.

Test Data First of all, we wanted to have relatively realistic data. To achieve this goal we used appraisal results of the SEI Appraisal Program [3]. SEI has designed "the Standard CMMI Appraisal Method for Process Improvement (SCAMP-ISM) to provide benchmark quality ratings relative to CMMI models" [4]. These results show the assigned maturity level of the staged version of CMMI-SW V1.1 model for the appraised organizations, based on the process areas determined by the appraisal method. In order to attain a maturity level, the organization should have "satisfied" or "not applicable" ratings for certain process areas, where the maturity level consists of these process areas. For our experiments, fifteen organizations were selected from the appraisal results. Appraisal results for the fifteen organizations and the ratings for the process areas applied by each organization are shown in Table 3. Table 4 shows the legend for Table 3.

In order to prepare data that can be processable by an OWL reasoner, we had to prepare annotations based upon the data shown in Table 3. Towards this aim, we had to create instances of various classes that 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 number of instances we had to create was quite high. For instance, for an organization to be at the maturity level 5, at least 249 instances need to be created. For other levels these numbers are: 231 for level 4, 213 for level 3 and 93 for level 2.

Testing First of all, we checked the consistency of the ontology using the ConsVISor consistency checker which is freely available at `http://www.vistology.com/consvisor`. The result of the test was that the ontologies were consistent.

In the next step we tried to derive maturity levels using an OWL inference engine. Our first attempt was to use two OWL inference engines that are available in the public domain - Pellet and Racer. However, we were not able to achieve the expected results. Both Pellet and Racer gave messages that we interpreted as "out of memory". Since we did not try to investigate the real causes of these errors, we are not making any claims about the real causes of such behaviors. At this time we did not have any other inference engine available, so we decided to scale down the CMMI-SW ontology and test the smaller version on Racer. All the test cases resulted in correct (expected)inference.

In the meantime, another inference engine, BaseVISor, became available to us. It is a prototype implementation and is not, as yet, available on the Internet. Information on BaseVISor can be found at `http://www.vistology.com`. BaseVISor was able to handle the test cases we developed and the results were as expected. In other words, for all the 15 cases, BaseVISor derived that the organizations satisfied the levels specified in Table 3, as well as all the levels below that highest level.

Table 3. Test cases

	O1	**O2**	**O3**	**O4**	**O5**	**O6**	**O7**	**O8**	**O9**	**O10**	**O11**	**O12**	**O13**	**O14**	**O15**
ApprR	3	5	3	5	5	3	2	4	2	4	2	3	4	2	5
OPF	S	S	S	S	S	S	NR	S	NR	S	NR	S	S	NR	S
OPD	S	S	S	S	S	S	NR	S	NR	S	NR	S	S	NR	S
OT	S	S	S	S	S	S	NR	S	NR	S	NR	S	S	NR	S
OPP	NR	S	NR	S	S	NR	NR	S	NR	S	NR	NR	S	NR	S
OID	NR	S	NR	S	S	NR	NR	NR	NR	NR	NR	NR	NS	NR	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	NA	S	S	S	NA	S	S	NA	S	NA	NA	NA	S
IPM	S	S	S	S	S	S	NR	S	NR	S	NR	S	S	NR	S
RSKM	S	S	S	S	S	S	NR	S	NR	S	NR	S	S	NR	S
IT	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
ISM	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
QPM	NR	S	NR	S	S	NR	NR	S	NR	S	NR	NR	S	NR	S
REQM	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
RD	S	S	NR	S	S	S	NR	S	NR	S	NR	S	S	NR	S
TS	S	S	S	S	S	S	NR	S	NR	S	NR	S	S	NR	S
PI	S	S	S	S	NA	S	NR	S	NR	S	NR	S	S	NR	S
VER	S	S	S	S	S	S	NR	S	NR	S	NR	S	S	NR	S
VAL	S	S	S	S	NA	S	NR	S	NR	S	NR	S	S	NR	S
CM	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
MA	S	S	S	S	S	S	S	S	S	S	S	S	S	S	S
DAR	S	S	S	S	S	S	NR	S	NR	S	NR	S	S	NR	S
OEI	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR	NR
CAR	NR	S	NR	S	S	NR	NR	NR	NR	NR	NR	NR	NS	NR	S

5 Related Work

Our literature search has not identified many efforts of implementing formal ontologies for software engineering in general and for CMMI-SW in particular. Mendes and Abran [5] have initiated a project for the development of a software engineering domain ontology based on the Software Engineering Body of Knowledge (SWEBoK). 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.

Liao, Qu and Leung [6] introduce an ontology named Software Process Ontology (SPO) to express software processes at the conceptual level. It has an extension to encapsulate specific process models such as CMMI and ISO/IEC 15504.

Our ontology for CMMI-SW, which uses rating and a sequence of improvements based on successive levels in the implementation of the field of software engineering, contrasted to the SWEBOK ontology of Mendes and Abran, which

Table 4. Legend for Test Results

Acronym	Description	Acronym	Description
AppR	Appraisal Result	OPF	Organizational Process Focus
OPD	Organizational Process Def.	OT	Organizational Training
OPP	Organizational Proc. Perf.	OID	Org. Innovation and Deplmnt
PP	Project Planning	PMC	Project Monitoring and Cntrl
SAM	Supplier Agreement Mngmnt	IPM	Integrated Project Mngmnt
RSKM	Risk Management	IT	Integrated Teaming
ISM	Integrated Supplier Mngmnt	QPM	Quality Project Mngmnt
REQM	Requirements Management	RD	Requirements Development
TS	Technical Solution	PI	Product Integration
VER	Verification	VAL	Validation
CM	Configuration Management	PPQA	Process & Prod. Qual. Assrn.
MA	Measurement and Analysis	DAR	Decision Analysis & Resol.
OEI	Organiz. Env. for Integr.	CAR	Causal Analysis & Resol.
S	Satisfied	NR	Not Rated
NA	Not Applicable	NS	Not Satisfied

uses a broad body of knowledge of the same field. The CMMI-SW ontology described in this paper is a comprehensive representation of the CMMI-SW model and thus is amenable to formal reasoning about levels of maturity of software organizations. This is a deviation from the broader ontology of Liao, Qu and Leung, which encapsulates other models in addition to CMMI.

6 Conclusions and Future Work

The main purpose of this implementation 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 CMMI-SW ontology was implemented in OWL-DL. The ontology includes 309 classes and four properties. 97 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 from developed by SEI Appraisal Program. In order to annotate these organizations, a large number of instances had to be added to the base CMMI-SW ontology. The number of instances ranged from 93 to 249, 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/. The particular test cases are identified as cmmi_test1.owl through cmmi_test15.owl.

Our initial attempts to use Pellet and Racer failed, perhaps due to the size of the ontology. But eventually the BaseVISor inference engine was used successfully and for all of the test cases the inference engine derived the same conclusions as in the appraisal results. Information on BaseVISor is available at: http://vistology.com.

Since the CMMI-SW 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.

Although the validation results indicate that the ontology faithfully captures the concepts and constraints of the CMMI-SW 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 are publishing the ontology on our web site. This paper is our first attempt to encourage the software engineering community to review the ontology and possibly send us comments so that it can be modified to more fully capture the ideas in the CMMI-SW model.

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

References

1. Capability maturity model integration (cmmism), version 1.1 cmmism for software engineering (cmmi-sw, v1.1) staged representation. Technical Report CMU/SEI-2002-TR-029, ESC/TR-2002-029, Carnegie Mellon, Software Engineering Institute, Pittsburgh, 2002.
2. W3C. Web Ontology Language Reference OWL, 2004. `http://www.w3.org/2004/OWL/`.
3. Carnegie mellon software engineering institute. list of published scampi appraisal results., 2005. `http://seir.sei.cmu.edu/pars/pars_list_iframe.asp`.
4. Carnegie mellon software engineering institute. cmmi appraisals., 2006. `http://www.sei.cmu.edu/cmmi/appraisals`.
5. O. Mendes and A. Abran. Software engineering ontology: A development methodology. *Metrics News*, 9:68–76, 2004.
6. L. Liao, Y. Qu, and H. Leung. A software process ontology and its application. In *ISWC2005 Workshop on Semantic Web Enabled Software Engineering*, 2005.