Gazi University Journal of Science
GU J Sci
25(1):155-164 (2012)

ORIGINAL ARTICLE

An Ontology Based Infrastructure To Support CMMI-Based Software Process Assessment

Sema GAZEL[1], Ebru AKÇAPINAR SEZER[1♠], Ayca TARHAN[1]

[1]Hacettepe University, Computer Engineering Department, Beytepe, Ankara,06800, Turkey

Received: 05.02.2011 Revised: 07.08.2011 Accepted: 16.10.2011

ABSTRACT
This paper presents an ontology-based software process assessment tool which was developed 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, is a realization of the process engineering meta-model SPEM. The study also explains findings from example usage of the OCMQT in a system and software development organization. In OCMQT, there is a need for expert knowledge. In fact, process improvement and assessment activities always require experts. However, OCMQT can protect non-expert personals from making unintended mistakes in an organization.

***Keywords**: Software Process Assessment Tool, CMMI, Ontology, SPEM, OWL.*

1. INTRODUCTION

Process improvement is a continuous activity and it carries out the Deming's plan-do-check-act steps[1] on an organization's process assets in a cyclic way. Definitions of process entities, setting the relationship between them, updating these definitions and relations when something is changed, and keeping the process assets under configuration control during process improvement, are important activities for organizations which adopted process oriented management [2]. Many times, it is almost a necessity to utilize supporting tools to perform process improvement activities.

Various works have been demonstrated for modeling and improving software processes during the last decades. Two of them include a meta-model for software and systems process engineering called SPEM [3], and a process reference model that addresses development and maintenance activities for both products and services called CMMI [4]. Software and System Process Engineering Meta-Model (SPEM) is a conceptual framework proposed by the Object Management Group (OMG) to define system and software processes and their components. There are tools developed by various organizations based on this meta-model, and one of them is Eclipse Process Framework (EPF) maintained by Eclipse Foundation [5]. EPF is an Eclipse-based, open-source software process engineering framework that can be extended by developing plug-ins. It contains a process management tool called EPF Composer [6] as an implementation of SPEM. EPF Composer enables definition of processes and methods as well as management, configuration, and deployment of libraries as related to software and system development. An organization can readily model its processes and manage these process models by using EPF Composer. Capability Maturity Model Integration (CMMI), on the other hand, is a process reference model developed by the Software Engineering Institute (SEI) of Carnegie Mellon University in order to guide development and maintenance activities applied to both products and services. CMMI can be used as reference

♠Corresponding author, e-mail: ebru@hacettepe.edu.tr while assessing the capability of a process or the maturity of an organization, and improving processes [4].

Process assessment is the foundation step for process improvement activities. It investigates strong, weak, and/or missing points in definition and application of a process in detail [7]. Process assessment provides an understanding about current process situation and enables rating about process quality based on this understanding. It can be performed by system/software engineering process group of the organization, by independent consultants, or a combination of both [4,7]. Findings from a process assessment are usually transformed into issues for process improvement. Process assessments, either performed internally or externally to the organization, are very time and effort consuming and require qualified personnel. Especially, if an organization tries to keep its organizational processes in accordance with more than one process reference model and/or standard; tracking compliance of the organization's processes to each model and/or standard and finding out the deviation between them would be getting more complicated. When it is done manually, these activities can be error-prone, since the environment is open to mistakes and difficult to manage. If process modeling tools could be extended with some abilities to carry out process assessment activities, process assessments would be supported and it would be beneficial to reduce assessment process costs and risks.

In order to provide such an infrastructure, it is required; 1) to formally represent process reference models so as to communicate with process management tools, 2) to map an organization's process assets residing in a process management tool to the formal representations of process reference models, and 3) to inquire these mappings for strengths and weaknesses. It is believed that the use of ontologies can be a good solution to meet these requirements. Ontologies enable to formally represent the knowledge in specific domains, to map one domain to another domain, and to inquiry this mapping. Because of these properties, using an ontology-based approach can support all three steps listed above.

Based on these assumptions, an ontology-based software process assessment tool was developed 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 EPF Composer. This study explains the tool and the findings from its example usage in a system and software development organization. The abilities of the OCMQT are as follows: 1) Viewing a CMMI ontology, 2) Creating ontology of a software process already represented in SPEM by EPF Composer, 3) Mapping (manually) the entities in software process ontology to the entities in CMMI ontology, and storing these mappings as ontology, 4) Querying CMMI ontology, software process ontology, and mapping ontology.

This study is organized as follows. Section 2 is divided into four subsections. Section 2.1 provides conceptual background on process improvement, CMMI, and process assessment. Section 2.2 explains the concepts of ontology and lists related languages and tools. Section 2.3 includes basic properties of SPEM and EPF. Section 2.4 provides a review of related literature on ontology-based process assessment environments. Section 3 explains the architecture and components of OCMQT as an ontological infrastructure to support CMMI-based software process assessment. Finally, section 4 provides overall conclusions.

2. BACKGROUND

2.1. Process improvement, CMMI, and process assessment

A process is a sequence of activities that transforms inputs into outputs to meet a purpose [8]. A software process is one or more processes that an organization or a project uses to plan, carry out, control, and improve software-related activities. Software process improvement is the process of changing an organization's software processes in order to meet the organization's business needs and to reach its business objectives more effectively [8]. Several process reference models have been demonstrated during the last decades for systematic improvement of software processes. Among these, "ISO/IEC 12207 Software Life Cycle Processes" [9] and "Capability Maturity Model Integration (CMMI) for Development" [4] have been utilized more extensively.

CMMI is a process reference model that was developed by the Software Engineering Institute (SEI) of Carnegie Mellon University. CMMI addresses the development and maintenance activities applied to both products and services. This model could be used for improving processes, and measuring the capability of a process or the maturity of an organization [4]. CMMI components (including a model, its training materials, and appraisal-related documents) are designed to meet the needs of some specific areas of interest, which is called constellation. There are three constellations as supported by the version 1.2 of the framework: CMMI for Development [4], CMMI for Services [10], and CMMI for Acquisition [11].

CMMI consists of process areas where domain related process activities are explained, goals and practices of these process areas, two different representations, and two different scopes. Representations are staged and continuous, and scopes are with IPPD (Integrated Product and Process Development) and without IPPD. The representations and the scopes indicate in what way the goals and practices shall be handled. The representations can be considered as two different viewpoints created by putting the model components together in two different ways. IPPD, which is an addition, enables expansion of the process areas in CMMI with goals and practices so as to cover the integrated team activities. Model components are grouped into three categories which are required, expected, and informative. Required and expected components are considered in process assessments, because they are important.

A process area is a cluster of related practices in an area that, when implemented collectively, satisfy a set of goals considered important for making improvement in that area. In CMMI for Development [8], all system and software development processes are addressed by 22 process areas such as Project Planning, Requirements Development, Measurement and Analysis, and etc; and these process areas are the same for both staged and continuous representations. Process areas are rated by maturity levels (ML1-ML5) to identify organizational maturity in the staged representation, whereas they are rated by capability levels (CL0-CL5) to identify process capability in the continuous representation. "5" means the most improved in both ratings. In both representations, each level constitutes a basis for the next level. Thus, in order to be considered successful in a level, it is required that the previous levels should be covered as well as those that are required to be covered at the level. The ratings of organizational maturity and process capability are not completely independent of each other. Because many model components are utilized in both representations, it is possible to map the ratings of process areas in one representation to the ratings in the other representation.

Process assessment is the systematic evaluation of an organization's processes against a process reference model [8]. There are several process assessment models, among which "Standard CMMI Appraisal Method for Process Improvement (SCAMPI)" [12] and "ISO/IEC 15504: Information Technology - Process Assessment" [7] have become more popular. SCAMPI is a process assessment method designed to evaluate conformance to CMMI models and to perform quality ratings accordingly.

There are many tools developed to support process assessment; and in a study that investigated 46 of them [13], Leung et al. [13] stated that the most of the tools utilize Microsoft Excel or Access as a base to store data. In this study, 8 tools are given in Table-1 with their descriptions. While the selection of the listed tools, tools which have automatic support for process improvement are chosen and also SPI partners are considered.

Table-1. A list of tools to support process assessment

Tool Name	Description
Appraisal Assistant [14]	This is a software application developed by Software Quality Institute of Griffith University to meet the requirements of CMMI appraisals and of ISO/IEC 15504. It supports assessment of organizational maturity and process capability. Models and mapping data are stored in a Microsoft Access database.
CMMiPal v1.0 [15]	This tool was developed by Chemuturi Consultancy. It enables manual mapping of an organization's processes to CMMI practices. Model and mapping data are stored in a Microsoft Access database.
CMM-Quest v1.2 [16]	This tool was developed by HM&S IT-Consulting to support preparation, data collection, data analysis and reporting phases of process assessments based on CMMI-Dev v1.2. It provides functionalities for selecting process areas and target levels as preparation, text-based screens for data collection, graphics for data analyses, and Microsoft Word and HTML based reporting facilities. It does not support modeling of organizational processes.
SPICE 1-2-1 [17]	This is a software tool developed by HM&S IT-Consulting to support process assessments in accordance with ISO/IEC 15504.
SPiCE-Lite Tool [18]	This tool was developed by HM&S IT-Consulting to assess conformance of organizational processes to ISO/IEC 15504 requirements. Assessment data are stored in a relational database.
Model Wizard [19]	This is a Windows-based application developed by Integrated System Diagnostics Incorporated. It enables users to store their process models in a relational databas .
Appraisal Wizard [20]	This is a Windows-based, client-server structured software product developed by Integrated System Diagnostics Incorporated. It is aimed to support management of planning, preparation, data collection, merging, and reporting activities as related to process assessments and process audits. It can co-operate with Model Wizard [80]. Data from assessments and audits are stored in a relational database
CMMI v1.2 Self Assessment Tool [21]	This is a Microsoft Excel based process assessment tool developed by Management Information Systems.

2.2. Ontology, languages, and tools

Ontologies are content theories about the sorts of objects, properties of objects, and relations between objects that are possible in a specified domain of knowledge [22]. Ontologies represented in a shareable ontology language enable formal representation of knowledge in a domain, and serve the achievement of interoperability in these systems. There is a need to perform some operations on ontologies utilized by applications. Basic operations are building, mapping, merging, and querying of ontologies. Since ontology servers are partially aware of the meaning behind formally represented data, they enable to query this data and to have inferences from it. Creating, mapping and querying of ontologies were achieved in this study.

There are a number of tools to manage ontologies [23]. Ontology development tools, for example, are utilized to create a new ontology or to modify an existing ontology. They also serve documentation, exportation, and importation of ontology. OntoEdit, Protégé, WebOde, and Ontolingua are the examples for ontology development tools [23,24]. Protégé (version 3.3.1) employed in this study is a free, open-source ontology editor and knowledge-base framework [25]. It 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.

Ontology languages have become popular by the introduction of the Semantic Web during the last decade. More recent ontology languages are XML ("eXtended Markup Language"), RDF ("Resource Description Framework"), KA2 ("Knowledge Annotation Initiative of Knowledge Acquisition"), SHOE ("Simple HTML Ontology Extension"), OIL ("Ontology Interchange Language"), DAML ("DARPA Agent Markup Language") and OWL ("Web Ontology Language") [24]. Ontologies created in this study were represented in OWL which is a markup language to publish and share ontologies on the Web. OWL was derived from DAML+OIL and structured over RDF, by World Wide Web Consortium (W3C)'s Web Ontology Working Group [26].

Ontologies formalized in ontology languages can be processed by logical reasoners to derive knowledge. Jena 2 developed by HP Laboratories; various reasoners for RDF, RDFS, and OWL; and RDQL as a flexible querying language; are examples to ontology query systems and languages. OWL-QL and RACER are other such systems. RACER uses nRQL as the query language, and supports reasoning in RDF, RDFS, and OWL. SPARQL, developed by W3C, is an RDF query language [27].

In this study, ontologies represented in OWL were queried by Jena. Jena is an open-source Java framework developed by HP Labs to build the Semantic Web. It provides an environment to utilize RDF, RDFS, OWL, and SPARQL, and includes a rule-based reasoning engine [28].

2.3. SPEM and EPF

Software and System Process Engineering Meta-Model (SPEM) [3] is a process engineering meta-model and a conceptual framework. It is proposed by the Object Management Group (OMG) and is based on Meta Object Facility (MOF) [29]. It is basically used for defining system and software processes and its components. SPEM is not a general purpose process modeling language. It includes elements to define any software or system development process and excludes objects specific to a certain domain or discipline (e.g. project management).

SPEM 2.0 elements are organized under seven packages and the meta-model is divided into logical units by this organization. The functions of the packages within the meta-model are briefly described below [3]. Although SPEM 2.0 provides recommendations on the implementation of these packages, it allows selection and implementation of packages for a specific need. Core package contains those meta-model classes and abstractions that build the base for classes in all other meta-model packages. Process Structure package defines the base for all process models and supports the creation of simple and flexible process models. Process Behavior package allows extending process structures with behavioral models. It does not define its own behavior modeling approach, but rather provides 'links' to existing externally-defined behavior models. Managed Content package introduces concepts for managing the textual content of development processes. These concepts can either be used standalone or in combination with process structure concepts. Method Content package provides the concepts to build up a development knowledge base that is independent of any specific processes and development projects. Processes reuse these method content elements and relate them into partially-ordered sequences that are customized to specific types of projects. Process with Methods package defines or redefines structures for integrating processes defined with Process Structure meta-model package concepts with instances of Method Content meta-model package concepts. Method Plugin package introduces concepts for designing and managing maintainable, large scale, reusable, and configurable libraries or repositories of method content and processes. These concepts allow arranging different parts of such a library based on different layers of concern similar to layered software architectures.

There are tools developed by various organizations based on this meta-model. As being implementations of SPEM, these tools can be utilized by process management staff of a development organization or a specific project to define and maintain processes. An implementation of this meta-model is Eclipse Process Framework (EPF) proposed by Eclipse Foundation [5]. EPF is an Eclipse-based and open-source software process engineering framework that can be extended by developing plug-ins. It contains a process management tool called EPF Composer as an implementation of SPEM and example processes definitions created by this tool. EPF Composer [6] is a process management tool that enables definition of processes and methods as well as management, configuration, and deployment of libraries as related to software and system development. The processes defined by using this tool can be deployed as web pages via the tool.

In this study, SPEM was utilized for the purpose of creating an organizational process ontology because of its flexibility and support of OMG which has wide-acceptance in specifications. A subset of elements within Process with Methods package in SPEM was chosen and an ontology called Process Structure Ontology (PSO) was created based on this subset. Because EPF Composer was extended by a plug-in developed in this study, SPEM is also indirectly utilized to define organizational processes. The ontologies of organizational processes were created based on Process Structure Ontology and by utilizing process definitions constructed by EPF Composer.

2.4.Ontology-Based Process Assessment Environments

There are lots of studies in the literature on the subjects of either process modeling or ontology. However, when searched for process modeling ontology as the intersection of these two subjects, the number of studies decreases significantly. In this section, several studies which are close to the purpose and scope of this study were elaborated.

There are only a few studies on CMMI ontology in the literature. Soydan et al. [30] presented OWL ontology for CMMI for Software Engineering v1.1. In this work, only staged representation was analyzed whereas in our study, it was aimed to meet the needs of both staged and continuous representations. Sharifloo et al. [31] introduced an ontology for CMMI for Acquisition v1.2. This ontology was based on SUMO [32] upper ontology using SOU-KIF [33] languages.

Rungratri and Usanavasin [34] proposed a framework called “CMMI v.1.2 based Gap Analysis Assistant Framework (CMMI-GAAF)” to perform automatic gap analysis with respect to CMMI. Within this framework, Project Assets Ontology (PAO), which was an ontology to merge CMMI process areas and project assets, was created. PAO was created based on CMMI ontology developed by Soydan et.al. [30]. Other units of the framework included Project Assets Repository (PAR) which was a storage to hold project assets, Project Assets Metadata Generator (PAM Generator) that merged the information in the PAR with PAO, Project Assets Metadata Repository which was a storage of project assets metadata, and Project Maturity Level Assessment (ProMLA) that performed assessment of project’s maturity level. PAM Generator created project assets metadata by using process assets and PAO and forwarded this metadata as input to process assessment.

It was stated in [34] that PAO includes generic and specific goals as well as generic and specific practices in CMMI; however, the way how generic goals and practices were handled was not clear in the study. In the present study, the objects under general and specific practices in the object hierarchy of CMMI were not included in CMMI ontology, and general and specific practices were directly mapped to process steps and artifacts. Furthermore, our study focused on mapping of organizational process assets whereas the study in [34] focused on mapping of project assets to CMMI components.

Liao et al. [35] aimed to create generic Software Process Ontology (SPO) and strived to ensure that it covered the requirements of both CMMI and ISO/IEC 15504. A process was represented by atomic practices in this study. It was stated in [35] 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. Liao et al. [35] targeted a generic software process ontology whereas our study aimed to develop ontology of CMMI for Development v1.2. In addition, organizational processes were represented in SPEM by using EPF Composer tool in our study rather that by using generic software process ontology.

Doheny and Filby [36] defined a conceptual process modeling framework and developed a tool to support modeling and assessment of software processes. The framework was based on Process Ontology (PO). It was stated in the study [36] that PO could be utilized to model software development processes as well as standards and best practices as related to software development. In PO, objects were represented under three categories which were artifacts, activities, and agents. Doheny ve Filby [36] aimed to develop a generic software process modeling process ontology as Liao et al. [35] did. Therefore, our study differs from [36] because of similar results stated above.

Garcia et al. [37] proposed a framework for integrated management of modeling and measuring of software processes. It is argued by this study that all models and meta models should be based on the same conceptualization of objects and relations for the sake of integration. The place of software process models within the conceptual architecture of the study was explained and Descriptive Process Modeling Ontology (SPMO) was introduced. Although it was stated that SPMO was developed based on SPEM, the details of the ontology were not explained. This study [37] was similar to our study in using SPEM as a reference to develop process ontology. However, the focus in our study was process assessment whereas the focus in [37] was process modeling and measurement.

Lee et al. [38] presented an ontology-based computational intelligent multi-agent system for CMMI assessment. The system could summarize evaluation reports by using agents and quality assurance ontology was built based on Process and Product Quality Assurance process area of CMMI. Lee et al. [39] also presented an Ontology-based Intelligent Decision Support Agent (OIDSA) to apply to Project Monitoring and Control process area of CMMI. In the present study, the intention was to cover all process areas in CMMI for Development v1.2, though not by representing the knowledge as specific to each process area but by considering the structure of the CMMI. Furthermore, in the studies [38] and [39], the ontologies were constructed according to a domain ontology structure. In other words, CMMI was used as an instance of the domain ontology. However, CMMI was selected as the domain in our study and ontology was constructed by considering CMMI concepts and relations only.

3. ONTOLOGY-BASED CMMI MAPPING AND QUERYING TOOL (OCMQT)

When models/standards are presented in ontology, they gain the abilities of machine process ability, share ability, and querying. Liao et al. [35] presents the advantages of ontology use for process modeling clearly. In addition to the advantages presented by Liao et al. [35], when both of process reference models/standards and organizational processes are represented by ontologies, they can be mapped to each other and queried. In Gazel et al. [40] the representation of CMMI with ontology is presented and also in this study, the tool called as OCMQT which uses CMMI Ontology[40] is presented as an helpful infrastructure for CMMI-based process assessment activities.

In the OCMQT, CMMI-Dev v1.2 was selected as the process reference model. SPEM presented by OMG is used as the meta-model for ontology derivation of organizational process definitions. EPF Composer is the realization of SPEM with the properties of an open source, eclipse based, extendible tool. The OCMQT was developed as plug-ins to EPF Composer. The main components of the OCMQT are illustrated in Figure 1 and described in the following section.

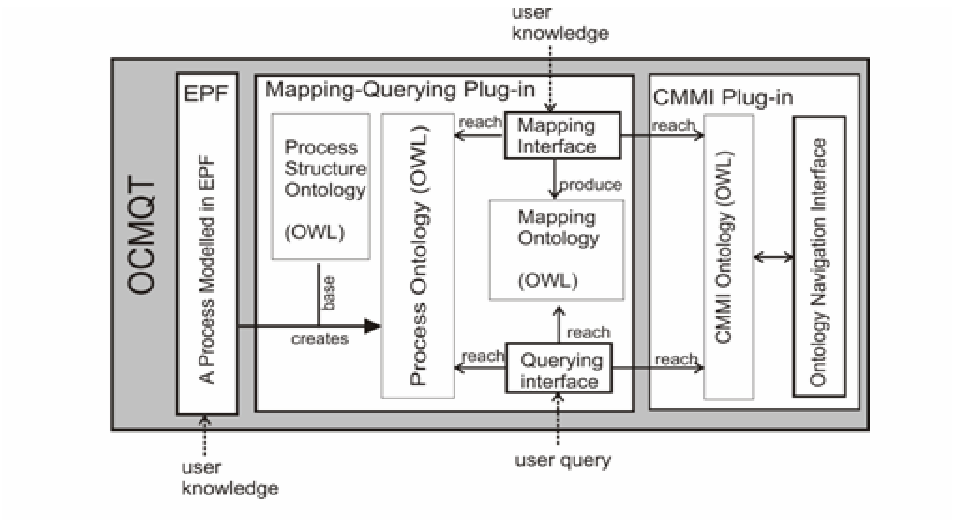

Figure 1: Schematic representation of OCMQT

- CMMI plug-in: This is a plug-in to view and manipulate CMMI Ontology in EPF Composer. First, CMMI-Dev v1.2 was modeled in an ontology and was represented in OWL by using Protégé-OWL Editor. Then, CMMI Ontology was integrated with EPF Composer (version 1.5) by using a software developed as an eclipse plug-in (version 3.4.1).
- Mapping-Querying plug-in: This is an eclipse plug-in (version 3.4.1) developed with the abilities listed below:
- Creating ontology of organizational processes already modeled in EPF Composer (Process Ontologies)
- Mapping of CMMI Ontology and Process Ontologies, and storage of mapping knowledge in another ontology (Mapping Ontology)
- Querying of CMMI Ontology, Process Ontologies, and Mapping Ontology, and listing of results

The major purpose of the CMMI Ontology design was to create a base for mapping of organizational process definitions and CMMI-Dev v1.2 components for supporting process assessment activities. The secondary aim of the design was to reflect CMMI domain knowledge as much as possible. When designing the CMMI Ontology model, continuous and staged representations were considered but Integrated Product and Process Development (IPPD) was excluded. First, ontology models for continuous and staged representations were designed individually and then, these models were combined by using common domain concepts. The rationale behind this combination is to provide the beneficial possibility of switching from one representation to the other. For example, it would be possible to see the corresponding levels of the staged and continuous representations. In Figure-2 [40], combined CMMI Ontology model for staged and continuous representations is illustrated with its concepts and relations.

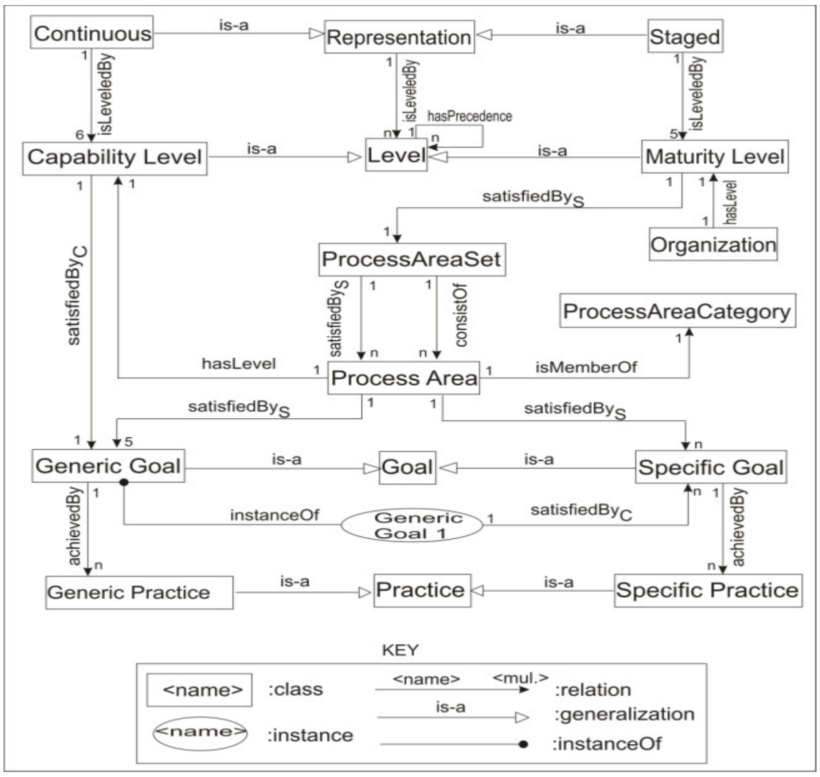

Figure-2: The Aggregated CMMI Ontology of staged and continuous representations [40]

Detailed explanation about Aggregated CMMI Ontology can be found in [40]. In summary, all classes reflects CMMI terms explicitly and the meanings of the relations are summarized from [40] and given in Table 2 .

Table-2: Meaning of relations in Aggregated CMMI Ontology

Name of the relation	Meaning
is-a	Inheritance
hasLevel	Denotes the organization level according to corresponding representation
isLeveledBy	Shows the levels of the corresponding representation
hasPrecedence	Reflects all previous levels that require to be satisfied
consistOf	Shows that which process areas constitutes the associated process area set
isMemberOf	Reflects the category of the process areas
satisfiedByS(taged) (towards ProcessAreaSet)	Denotes the process area set that should be satisfied to accept associated maturity level as satisfied
satisfiedByS(taged) (towards ProcessArea)	Reflects all process areas that should be satisfied to accept the process are set as satisfied
satisfiedByS(taged) and satisfiedByC(ountinuous) (towards Goals)	List of the goals that require to be satisfied
achievedBy	Reflects the satisfactory practices of the Goal
instanceOf	Reflects the instance of the class

In EPF Composer, processes are expressed with the object types such as Delivery Process and Capability Pattern. Capability Pattern is the structure containing the part of the process. Reusable process modules can be constituted with this structure. Whole process is defined with Delivery Process object type. In other words, one organizational process or the process applied in a project can be defined with Delivery Process object. In this study, Process Structure Ontology was organized based on Delivery Process object. Delivery Process object includes hierarchical structure and relations as explained below:

- In EPF, a Delivery Process object includes the objects such as Phase, Iteration, Activity, Milestone, Task Descriptor, Work Product Descriptor, Role Descriptor, and Team Profile.
- Delivery Process, Capability Pattern, Phase and Iteration objects are derived from Activity object. In hierarchical structure, these objects can cover other objects that are covered by Delivery Process object.
- Milestone, Task Descriptor, Work Product Descriptor, and Role Descriptor are leaf objects of the hierarchical structure and they cannot cover any objects.
- Team Profile object can cover only Team Profile object.

The includes relation was created specifically in Process Structure Ontology to show the hierarchical structure between the classes covered in a Delivery Process. When ontology of an organizational process definition is being created, the classes Activity, Delivery Process, Iteration, Milestone, Phase, Role Descriptor, Task Descriptor, Team Profile, and Work Product Descriptor are utilized. All of these classes and their relations correspond to a subset of the classes and the relations placed in the package called Process with Methods in SPEM.

Process Ontology is the ontology of an organizational process already modeled in EPF Composer. Typically, more than one organizational process are modeled in EPF Composer. For this reason, the ontology of each process is stored in a different OWL file, and each ontology uses the Process Structure Ontology as a super ontology. While Process Structure Ontology includes all classes and relations to model a process; in Process Ontology, the universal unique descriptors of the process assets modeled in EPF Composer is placed. Objects of the Process Ontology is created based on the classes and relations of the Process Structure Ontology. For example, assume that "do unit test" activity is placed in a process. Activity class of the Process Structure Ontology is used as the base class for Process Ontology objects which cover "do unit test" activity and its universal unique descriptor.

Mapping Ontology uses the CMMI Ontology and Process Ontologies, and stores the information of mappings between CMMI components and process assets. In fact, these mappings is many to many (m-n). In other words, one CMMI component can correspond to many process asset or vice versa. In practice, mapping is expected between CMMI practices and process activities, tasks, or work products. However, any CMMI component can be mapped with any process asset in the infrastructure presented in this study. The relations between ontologies presented above are illustrated in Figure-3.

o

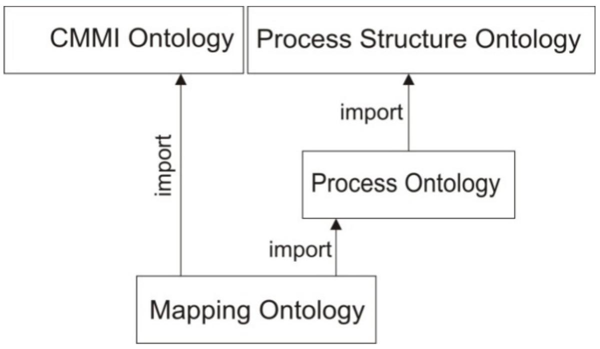

Figure 3: The relations between ontologies utilized in OCMQT

Basic facilities of the OCMQT are listed following and a snapshot from the OCMQT is given in Figure-4.

- The creation of the process ontology selected from Delivery Process in the library view,
- Examination of the process and all assets of it in a process editor window,
- Establishing the mapping between selected components from CMMI Ontology and the process ontology,
- Deleting/updating/listing of mappings,
- Loading/saving ontologies of CMMI, process, and mapping.

Querying abilities supported with OCMQT are listed below:

- Querying about CMMI domain knowledge: It is possible to list generic practices, specific practices, generic goals, and specific goals for a selected process area. In addition, explanations are displayed on the tool for practices and goals when they are selected.
- Querying about process: It is possible to list all activities, tasks, roles, and work products for a selected process within all defined processes.
- Querying about mapping: The mappings between CMMI components and processes have many-to-many (m:n), bidirectional multiplicity; accordingly, there are two types of querying such as;
  - Listing of process assets related with the selected CMMI component
  - Listing of CMMI components related with the selected process asset

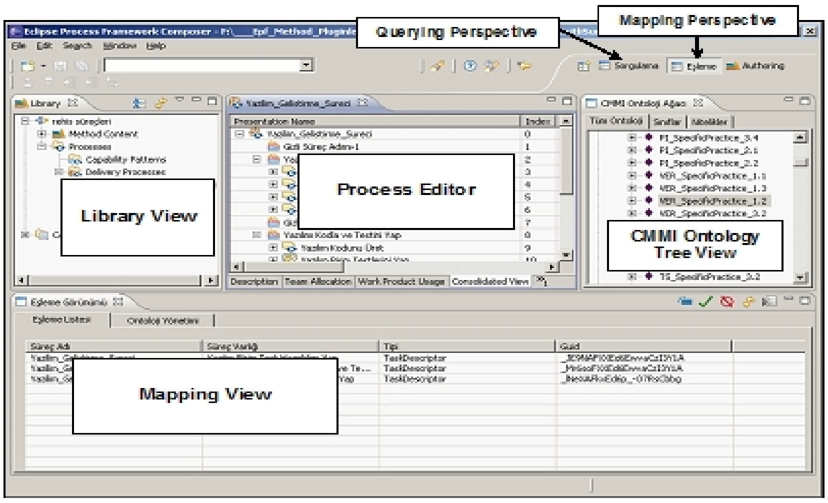

Figure-4: A snapshot from the ontological infrastructure of OCMQT

OCMQT was utilized in a system and software development organization which readily used Microsoft Word to maintain process definitions and Microsoft Excel to store mappings between process assets and CMMI components. Observations can be summarized like that establishment and management of mapping between CMMI components and organizational process assets was performed easier, faster, and more accurately with OCMQT than Microsoft Excel. The use of OCMQT eliminated the need for searching on process definitions and CMMI documents again and again. At the beginning of QCMQT usage, it seemed as a time consuming effort to define whole organizational processes in detail by using OCMQT infrastructure. However, after completing this step, usefulness of OCMQT became evident, especially in understanding CMMI knowledge, accessing process assets, and managing mappings for process assessment and improvement.

4. CONCLUSION

OCMQT enables an integrated infrastructure for process improvement and CMMI based assessment activities. It tries to map definitions of organizational process and CMMI. It uses the SPEM as the metamodel for organizational process and EPF as the realization of the SPEM. It uses the ontology technology to express these definitions and also gains the all advantages of the ontologies. In OCMQT, user interfaces are supplied for mapping, namely, user can select the CMMI component and process asset from different frames and maps them easily. All of the mappings are stored with directional relation to enable effective querying.

When a modification is considered for a process, the effect of the modification on the consistency with CMMI, lacks and errors of the process can be seen easy with the OCMQT. Moreover, data repetitions are eliminated because of the integrated infrastructure. Presently, OCMQT does not have the ability of inference about assessment rating. However, OCMQT gives an infrastructure for the design and implementation of advanced features. Although there is a need for expert knowledge for mapping, this necessity does not arise with the use of OCMQT. In fact, process improvement and assessment activities always require experts. However, OCMQT can protect non-expert personnel from making unintended mistakes in an organization.

5. REFERENCES

[1] Tague, N.R., “The Quality Toolbox”, 2nd Ed., ***ASQ Quality Press***, ISBN 978-0-87389-639-9: 390-392 (2004).

[2] “Quality Management Systems – Requirement”, ***ISO, ISO 9001:2000*** (2000).

[3] “Software & Systems Process Engineering Metamodel Specification (SPEM)”, version 2.0, OMG Document Number: formal/2008-04-01, Object Management Group (OMG), (2008).

[4] CMMI Product Team, CMU/SEI. CMMI for Development. CMMI-SE/SW V1.2, CMU/SEI-2006-TR-008, ESC-TR-2006-008, August (2006).

[5] “Eclipse Foundation”, http://www.eclipse.org

[6] Haumer, P., “Introducing the Eclipse Process Framework”, ***Eclipse Con,*** (2006).

[7] “Information Technology - Process Assessment”, ISO, ISO/IEC 15504 (2003-2008).

[8] “Information technology — Software process assessment” —Part 9:Vocabulary, First edition, ISO/IEC, 15504-9, (1998-08-15).

[9] “Systems And Software Engineering” – Software Life Cycle Processes**,** ***IEEE Std*** 12207-2008, Second Edition, ISO. ISO/IEC 12207 (01-02-2008).

[10] CMMI Product Team, CMU/SEI. CMMI for Services. CMMI-SVC V1.2, CMU/SEI-2009-TR-001, ESC-TR-2009-01, February 2009.

[11] “CMMI Product Team, CMU/SEI. CMMI for Acquisition”,CMMI-ACQ V1.2, CMU/SEI-2007-TR-017, ESC-TR-2007-017, November (2007).

[12] "SCAMPI Upgrade Team. Standard CMMI Appraisal Method for Process Improvement (SCAMPI) A, Version 1.2", ***Method Definition Document, Handbook,*** CMU/SEI-2006-HB-002, August (2006).

[13] Leung H.K.N., Liao L., Qu Y.. Automated support of software quality improvement. International Journal of Quality & Reliability Management, 24:(3):230-243 (2007).

[14] Appraisal Assistant, "Griffith University", ***Software Quality Institute.*** http://www.sqi.gu.edu.au/AppraisalAssistant/about.html

[15] http://www.chemuturi.com/cmmipaldtls.html

[16] http://www.cmm-quest.com

[17] http://www.spice121.com

[18] http://www.spicelite.com

[19] Model Wizard. http://isd-inc.com/tools.modelWizard

[20] Appraisal Wizard. http://isd-inc.com/tools.appraisalWizard

[21] http://www.man-info-systems.com/index_files/FreeTools.htm

[22] B. Chandrasekaran, J.R. Josephson, V.R. Benjamins, "What Are Ontologies, and Why Do We Need Them?", ***IEEE Intelligent Systems*** 14(1):20-26 (1999).

[23] OntoWeb Consortium. Deliverable 1.3: "A survey on ontology tools. OntoWeb Ontology-based information exchange for knowledge management and electronic commerce", IST-2000-29243. May, (2002).

[24] Pulido, J.R.G., Ruiz, M.A.G., Herrera, R., Cabello, E., Legrand, S., Elliman, D., "Ontology languages for the semantic web: A never completely updated review". ***Knowledge-Based Systems***, November, 19 (7): 489–497 (2006).

[25] Protégé. http://protege.stanford.edu

[26] W3C. Owl Web Ontology Language Reference. Technical Report, February 2004.

[27] Z. Zhang. "Ontology Query Languages For The Semantic Web: A Performance Evaluation", ***A Master of Science Degree Thesis, The University Of Georgia,*** Athens, (2005).

[28] Jena. http://jena.sourceforge.net

[29] OMG, Object Management Group. http://www.omg.org

[30] G.H. Soydan, M.M. Kokar. "An OWL Ontology for Representing the CMMI-SW Model. In ISWC 2006" ***Workshop on Semantic Web Enabled Software Engineering***, 2006.

[31] Sharifloo A.A., Motazedi Y., Shamsfard M., Dehkharghani R.. "An Ontology for CMMI-ACQ Model", ***In 3rd International Conference on Information and Communication Technologies: From Theory to Applications (ICTTA),*** April (2008).

[32] "Suggested Upper Merged Ontology (SUMO)", http://www.ontologyportal.org

[33] "Knowledge Interchange Format, draft proposed" ***American National Standard*** (dpANS), NCITS.T2/98-004, Ed.

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

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

[36] Doheny, J. G., Filby, I. M., "A Framework and Tool for Modelling and Assessing Software Development Processes", ***In The European Software Control and Metrics Conference, Wilmslow***, May (1996).

[37] García, F., Piattini, M., Ruiz, F., Canfora, G., Visaggio C. A., "FMESP: Framework for the Modeling and Evaluation of Software Processes", ***Journal of Systems Architecture: the EUROMICRO Journal,*** 52 (11) : 627–639, November (2006).

[38] Lee, C.S., Wang, M.H., "Ontology-based Computational Intelligent Multi-agent and Its Application to CMMI Assessment", ***Springer Science+Business Media***, LLC (2007).

[39] Lee, C.S., Wang, M.H., Chen, J.-J., Hsu, C.Y., "Ontology-based Intelligent Decision Support Agent for CMMI Project Monitoring and Control", ***International Journal of Approximate Reasoning,*** April, 48: 62–76 (2008).

[40] Gazel, S., Tarhan, A., Sezer, E., "A CMMI Ontology for An Ontology-Based Software Process Assessment Tool", ***In proceedings of the 16th EuroSPI2 Conference***, 9.1 (2009).