Appl Intell (2009) 30: 203–219
DOI 10.1007/s10489-007-0071-1

Ontology-based computational intelligent multi-agent and its application to CMMI assessment

Chang-Shing Lee · Mei-Hui Wang

Published online: 6 July 2007

Abstract This study presents an ontology-based computational intelligent multi-agent system for Capability Maturity Model Integration (CMMI) assessment. An ontology model is developed to represent the CMMI domain knowledge that will be adopted by the computational intelligent multi-agent. The CMMI ontology is predefined by domain experts, and created by the ontology generating system. The computational intelligent multi-agent comprises a natural language processing agent, an ontological reasoning agent and a summary agent. The multi-agent deals with the evaluation reports from the natural language processing agent, infers the term relation strength between the ontology and the evaluation report, and then summarizes the main sentences of the evaluation report. The summary reports are meanwhile transmitted back to the domain expert, which makes the domain expert further adjust the CMMI ontology. Experimental results indicate that the ontology-based computational intelligent multi-agent can effectively summarize the evaluation reports for the CMMI assessment.

Keywords Ontology · Intelligent multi-agent · CMMI · Fuzzy inference · Summarization

1 Introduction

Capability Maturity Model Integration (CMMI) is composed of best practices that consider the development and maintenance of products and services covering the product life cycle from conception through delivery and maintenance. By integrating these bodies of knowledge, which are essential for developing products, CMMI provides a comprehensive solution for developing and maintaining products and services [19]. This study describes one of the results for the four-year project, namely the CMMI Assistant with Service-oriented Information Marketplace (SIM), sponsored by Ministry of Economic Affairs of Taiwan from 2004 to 2007. Figure 1 shows the architecture of the CMMI Assistant with SIM.

According to the continuous representation, the process areas of the CMMI Assistant with SIM are divided into four categories, namely process management, project management, engineering, and support. The project management category has three web services, namely project closure (PC), project planning (PP) and project and monitoring control (PMC) services. The engineering category contains the requirements management (REQM), validation and verification (V&V), requirements development (RD), technical solution (TS) and model-driven architecture supporting (MDAS) services. The organizational process definition (OPD) services and the organizational process focus (OPF) services are the main functions of the process management category. Finally, the measurement and analysis (MA), configuration management (CM) and process and product quality assurance (PPQA) services are developed for the support category.

The purpose of process and product quality assurance process area is to provide staff and management with an objective perspective into the quality assurance of processes and related work products [19]. Quality assurance, or QA, can be defined in various ways. QA is most commonly defined as guaranteeing that developers, testers or independent auditors have performed scrutiny on a system to ensure that

C.-S. Lee (✉) · M.-H. Wang
Department of Computer Science and Information Engineering,
National University of Tainan, Tainan 700, Taiwan
e-mail: leecs@mail.nutn.edu.tw

M.-H. Wang
e-mail: mh.alice.wang@gmail.com

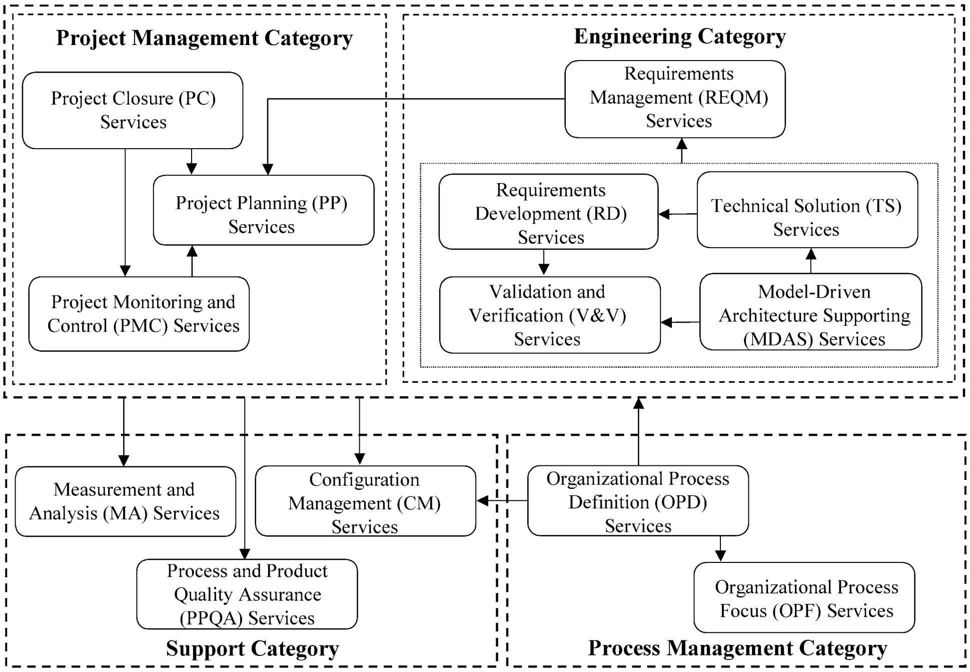

Project Management Category
Project Closure (PC) Services
Project Planning (PP) Services
Project Monitoring and Control (PMC) Services
Engineering Category
Requirements Management (REQM) Services
Requirements Development (RD) Services
Technical Solution (TS) Services
Validation and Verification (V&V) Services
Model-Driven Architecture Supporting (MDAS) Services
Measurement and Analysis (MA) Services
Configuration Management (CM) Services
Process and Product Quality Assurance (PPQA) Services
Support Category
Organizational Process Definition (OPD) Services
Organizational Process Focus (OPF) Services
Process Management Category

Fig. 1 Architecture of the CMMI Assistant with SIM

it works as required [25]. Many reports are often produced when developing and managing projects, wasting much effort and money. Therefore, noncompliance needs to be identified in the early development phases, and the relevant stakeholders must be able to acquire main sentences during assessment. This study presents an ontology-based computational intelligent multi-agent for CMMI assessment. The computational intelligent multi-agent comprises a natural language processing agent, an ontological reasoning agent and a summary agent. The CMMI ontology is predefined by domain experts, and generated by the ontology generating system. Based on the CMMI ontology and the Chinese Dictionary, developed by the Chinese Knowledge Information Processing (CKIP) group, the natural language processing agent handles the evaluation report, and filters insignificant terms from it. The ontological reasoning agent then infers the term relation strength of term pairs, and the summary agent summarizes the key features of the evaluation report. The project manager, the PPQA team members, and the project members can then retrieve the summarized evaluation reports to determine the exact level of quality achieved, and whether the processes and products function as required. All of the summarized results are recorded in the quality assurance repository, and sent back to the domain experts, allowing further modification of the CMMI ontology. Experimental results reveal that the proposed ontology-based computational intelligent multi-agent can effectively summarize the evaluation reports.

The remainder of this paper is organized as follows. Section 2 briefly introduces the related work of the agent, the ontology and CMMI applications. Section 3 then describes the structure of the ontology-based computational intelligent multi-agent. Next, Sect. 4 presents the structure of the CMMI ontology. Section 5 introduces the proposed computational intelligent multi-agent for CMMI assessment. The experimental results are shown in Sect. 6. Conclusions are finally drawn in Sect. 7, along with recommendations for future research.

2 Related work

An agent is a physical or virtual entity that can act in an environment and communicate directly with other agents. The concept of action is based on the fact that the agents conduct actions that modify the agents' environment and future decision-making [1, 2]. A multi-agent system consists of a number of agents interacting with each other. In the most general case, agents act on behalf of users with different goals and motivations. To interact successfully, agents need the ability to cooperate, coordinate and negotiate with one another, much as people do. The research topics in multi-agent systems include cooperation and coordination, communication, negotiation and scientific communities [26]. An agent can be defined in many ways. For instance, an agent possesses skills; can offer service; is capable of perceiving its environment, or is driven by a set of tendencies [2]. Furthermore, an intelligent agent is more powerful than an agent due to its reasoning and learning capabilities [3]. Delen et al. [4] designed and developed an intelligent decision support system for manufacturing to improve decision-making by managers for increasingly complex problem scenarios. Soo et al. [5] presented a cooperative multi-agent platform to help industrial knowledge managers retrieve and analyze existing patent documents, and extract structured information from patents with the aid of ontology and natural language processing techniques. Hamdi [6] proposed a multi-agent customization system based on machine learning mechanism for web mining. Lee et al. [7] developed a genetic fuzzy agent for meeting-scheduling systems. Wang [8] applied agent-based control for networked systems in control theory to traffic and transportation management. Yan et al. [9] developed a perceptron-based medical decision support system for diagnosing heart diseases. Grossklags et al. [10] studied the impact of software agents on the market behavior of human traders.

An ontology is a computational model of some portions of the world. It is a collection of key concepts and their inter-relationships, collectively giving an abstract view of an application domain [1]. The ontology aims at interweaving human understanding of symbols with their machine processability. Additionally, ontologies are shareable and reusable, are no longer of interest only in research [16]. The relevance of ontologies has been recognized in several practical fields, such as natural language translation, geographic information systems, biology and medicine, agent systems, knowledge management systems and e-commerce platforms. For example, Alani et al. [11] presented an automatic ontology-based knowledge extraction from Web documents, and adopted it to create personalized biographies. Chau [12] proposed an ontology-based knowledge management system to assist engineers in sharing, searching and managing knowledge on flow and water quality modeling. Corby et al. [13] presented an ontology-based search engine for the semantic web. Navigli et al. [14] developed an OntoLearn system for automatic ontology learning. Morbach et al. [15] reported on the development and application of a large-scale ontology for chemical process engineering. Francisio et al. [16] developed an ontology-based intelligent web portal system for recruitment tasks. They utilized an ontology to represent the knowledge of the recruitment domain. Burstein [17] presented a dynamic invocation of semantic web services using unfamiliar ontologies. Lee et al. [1] presented a fuzzy ontology, and applied it to news summarization. They also presented [18] a novel episode-based ontology construction mechanism to extract domain ontologies from unstructured text documents.

Capability Maturity Model Integration (CMMI) is a model for process improvement, and provides an opportunity to avoid or eliminate the bottlenecks and barriers that exist in organizations through integrated models that transcend disciplines [19]. CMM/CMMI has been widely researched. For instance, Staples et al. [20] studied why organizations do not use CMMI. The most frequent reasons given by organizations were as follows: the organization is small; the services are too costly; the organization has no time, and the organization is using another Software Process Improvement (SPI) model. Huang et al. [21] presented a decision support model to assist managers in identifying the priorities of the CMMI process areas. As software projects have become increasingly large and complex, a controlled development process with defined stages, estimation and measurement of the resources and the effort involved is increasingly considered to be necessary. However, software projects are often over budget, behind schedule and of poor quality. The process and product quality assurance is very important for improving the quality of the products [22]. Consequently, Liu et al. [23] discussed design, implementation and evaluation of an experimental intelligent software early warning system based on fuzzy logic using an integrated set of software metrics. Such a system can evaluate the quality, schedule and budget risks. Grief [24] presented a software testing and preventive quality assurance system for metrology, designed to increase orientation towards preventive quality assurance for the development processes, and to ensure that the product tests are repeatable and comparable.

3 Ontology-based computational intelligent multi-agent

The software is not just the program, but also all associated documentations and configuration data that are required to make these programs operate correctly. A software system generally comprises a number of separate files, including programs, configuration files, system documents and user documents [22]. This section introduces the system architecture and the computational intelligent multi-agent. Figure 2 shows the system architecture of the ontology-based computational intelligent multi-agent for CMMI assessment.

The domain experts predefine the domain knowledge about CMMI. The CMMI ontology is then constructed by the ontology generating system. The PPQA team members follow the product-based process quality assurance to evaluate the process and product objectively, and to identify and record the noncompliance issues into the evaluation reports. Based on the Chinese Dictionary [1] and the pre-constructed CMMI ontology, the computational intelligent multi-agent proceeds with the natural language processing agent, term relation strength reasoning and document summarization to generate the summarized evaluation report. The summarized report is then stored in the PPQA report repository. The key features of the evaluation report enable the project manager and project members to know precisely whether anything needs to be corrected for the ongoing project. Additionally, the PPQA team members, project manager and project members can retrieve the information stored in the PPQA report repository through the PPQA reporting system. Finally, the information presented on the PPQA reporting

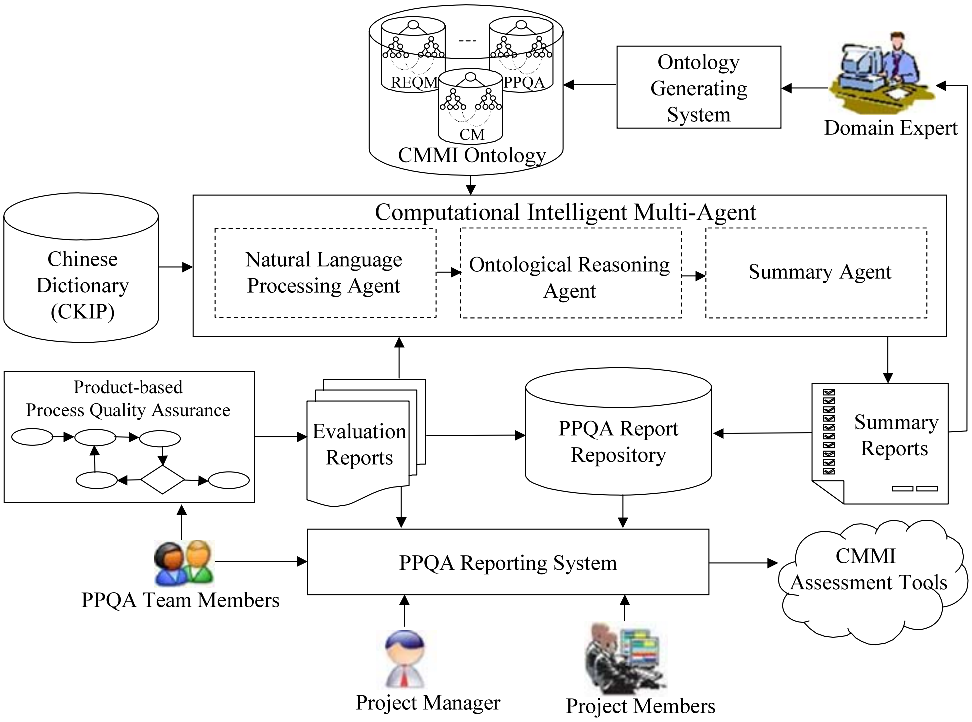


Fig. 2 System architecture of a computational intelligent multi-agent for CMMI assessment

system can support the CMMI assessment tools. Definitions for the ontology-based computational intelligent multi-agent are given as follows.

Definition 1 An *Ontology-based Computational Intelligent Multi-Agent* (*OCIMA*) is a multi-agent with a domain ontology *DO*, a Chinese Dictionary provided by the Chinese Knowledge Information Processing (CKIP) group CD_{CKIP}, an evaluation report *ER*, and a set of fuzzy inference rules *FIR*. The *OCIMA* can generate a summary report *SR*, that is,

$$OCIMA(CD_{\mathrm{CKIP}}, ER, DO; FIR) \mapsto SR.$$

Definition 2 A *Natural Language Processing Agent* (*NLPA*) receives a Chinese Dictionary CD_{CKIP} and an evaluation report *ER*, and can output a term set T_{NLPA} for the evaluation report *ER*, that is,

$$NLPA(CD_{\mathrm{CKIP}}, ER) \mapsto T_{\mathrm{NLPA}}.$$

Definition 3 An *Ontological Reasoning Agent* (*ORA*) is an agent with an outputting term set of Natural Language Processing Agent (*NLPA*) T_{NLPA} for the evaluation report *ER*, a set of ontological concepts C_O for the Domain Ontology *DO* and a set of fuzzy inference rules *FIR*, and can generate a set of term relation strength *TRS*, that is,

$$ORA(T_{\mathrm{NLPA}}, C_O; FIR) \mapsto TRS.$$

Definition 4 A *Summary Agent* (*SA*) is an agent with a domain ontology *DO*, a set of matched ontological concepts C_{mO} for the Domain Ontology *DO*, a matched term set T_{mNLPA} for the evaluation report *ER*, and a set of fuzzy inference rules *FIR*, that can create a summary report *SR*, that is,

$$SA(T_{\mathrm{mNLPA}}, C_{mO}, DO) \mapsto SR.$$

Property 1 The *Ontology-based Computational Intelligent Multi-Agent* (*OCIMA*) comprises a *Natural Language Processing Agent* (*NLPA*), an *Ontological Reasoning Agent* (*ORA*), and a *Summary Agent* (*SA*), that is,

$$OCIMA(\cdot) \leftarrow SA(\cdot) \bullet ORA(\cdot) \bullet NLPA(\cdot).$$

4 The structure of CMMI ontology

This study presents an ontology-based computational intelligent multi-agent based on a PPQA ontology model for supporting CMMI assessment. The structure and definition for the domain ontology are described below.

4.1 Ontology structure

Figure 3 shows the structure of the domain ontology adopted in this study, including the *domain layer*, the *category layer*, and the *concept layer* [1]. The *domain layer* represents the domain name of an ontology, and consists of various categories defined by domain experts. The *category layer* defines several categories, labeled as "category 1, category 2, category 3, …, category *k*". Each concept in the

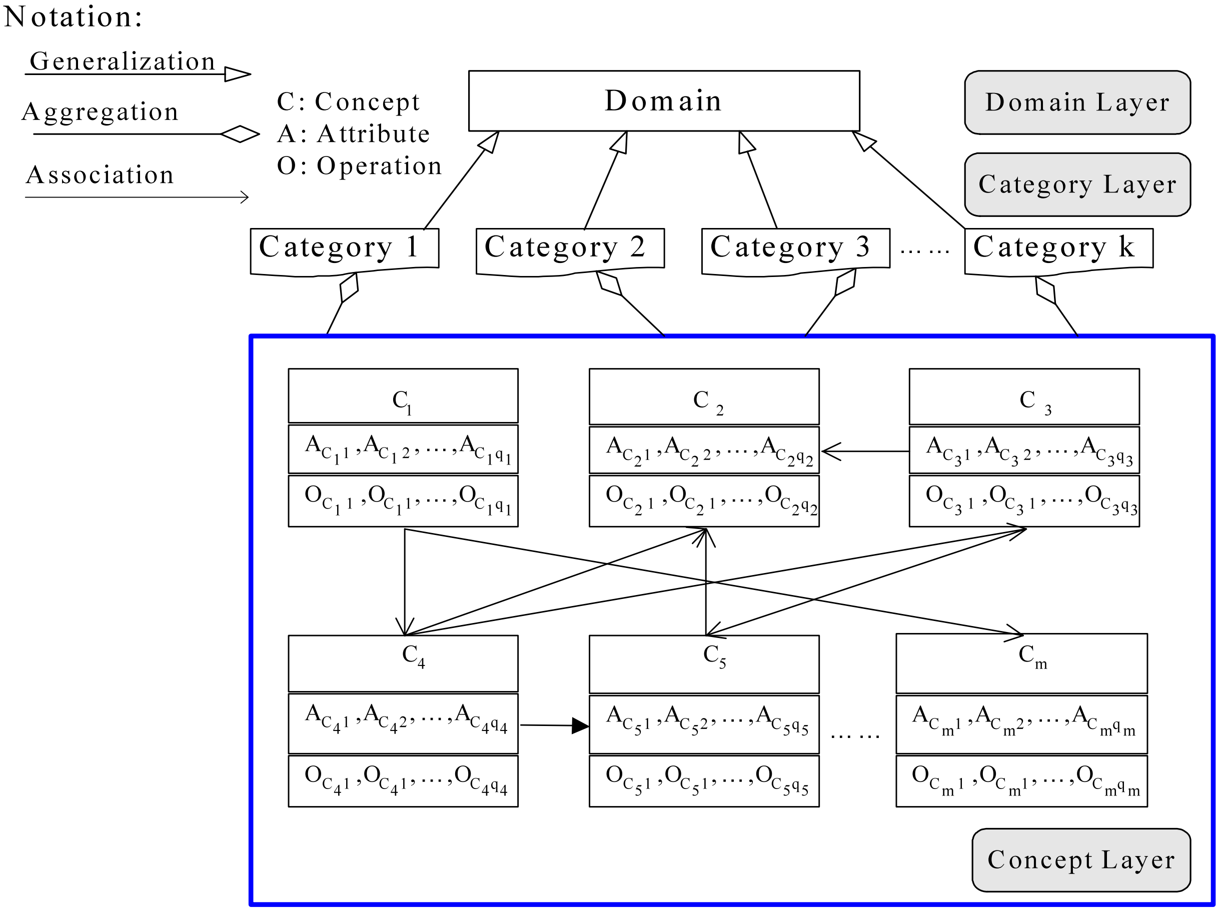

Notation:
Generalization
Aggregation
Association
C: Concept
A: Attribute
O: Operation
Domain
Domain Layer
Category Layer
Category 1
Category 2
Category 3
……
Category k
C_1
$A_{C_1 1}, A_{C_1 2}, \dots, A_{C_1 q_1}$
$O_{C_1 1}, O_{C_1 1}, \dots, O_{C_1 q_1}$
C_2
$A_{C_2 1}, A_{C_2 2}, \dots, A_{C_2 q_2}$
$O_{C_2 1}, O_{C_2 1}, \dots, O_{C_2 q_2}$
C_3
$A_{C_3 1}, A_{C_3 2}, \dots, A_{C_3 q_3}$
$O_{C_3 1}, O_{C_3 1}, \dots, O_{C_3 q_3}$
C_4
$A_{C_4 1}, A_{C_4 2}, \dots, A_{C_4 q_4}$
$O_{C_4 1}, O_{C_4 1}, \dots, O_{C_4 q_4}$
C_5
$A_{C_5 1}, A_{C_5 2}, \dots, A_{C_5 q_5}$
$O_{C_5 1}, O_{C_5 1}, \dots, O_{C_5 q_5}$
……
C_m
$A_{C_m 1}, A_{C_m 2}, \dots, A_{C_m q_m}$
$O_{C_m 1}, O_{C_m 1}, \dots, O_{C_m q_m}$
Concept Layer

Fig. 3 Structure of the domain ontology

concept layer, contains a concept name C_i, an attribute set $\{A_{C_i 1}, \dots, A_{C_i q_i}\}$, and an operation set $\{O_{C_i 1}, \dots, O_{C_i q_i}\}$ for an application domain. Three inter-conceptual relations have been utilized in the domain ontology, namely the *generalization*, the *aggregation*, and the *association*. A *generation* relation between a domain and its corresponding category is called an "*is-kind-of*" relationship. The relationship between each category and its corresponding concepts is the *aggregation*, which denotes the "*is-part-of*" relationship. The *association* indicates a semantic relationship between concepts in the *concept layer* [1].

4.2 PPQA ontology

Since software is increasingly becoming a larger part of many products and services, and the quality of a system is highly influenced by the quality of the process, a well-defined software development process plays a large role in determining the quality of a system. A CMMI is a reference model of mature practices in a specified discipline, and is employed to enhance and appraise a group's capability to perform that discipline. Additionally, a CMMI model provides guidance when developing or enhancing the organization's processes, and the ability to manage the development, acquisition and maintenance of products or services [19]. The CMMI process areas at maturity level 2 are 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), and Configuration Management (CM). This study builds a PPQA ontology based on the PPQA process area of CMMI. The role of the PPQA group within in process improvement is to evaluate objectively the adherence of the performed processes and associated work products and services to applicable processes descriptions, standards and procedures, in order to identify and document noncompliance issues in the evaluation reports.

This study designs the ontology for the PPQA domain based on the fundamental knowledge of the PPQA process area of CMMI [19] and the ontology definition shown in Fig. 3. The *concept layer* is constructed according to the five sub-layers, namely the *who*, *when*, *what*, *where*, and *how layers*. Figure 4(a) shows the structure of the partial PPQA ontology, and Fig. 4(b) displays its corresponding Chinese version. In the *domain layer*, the domain name

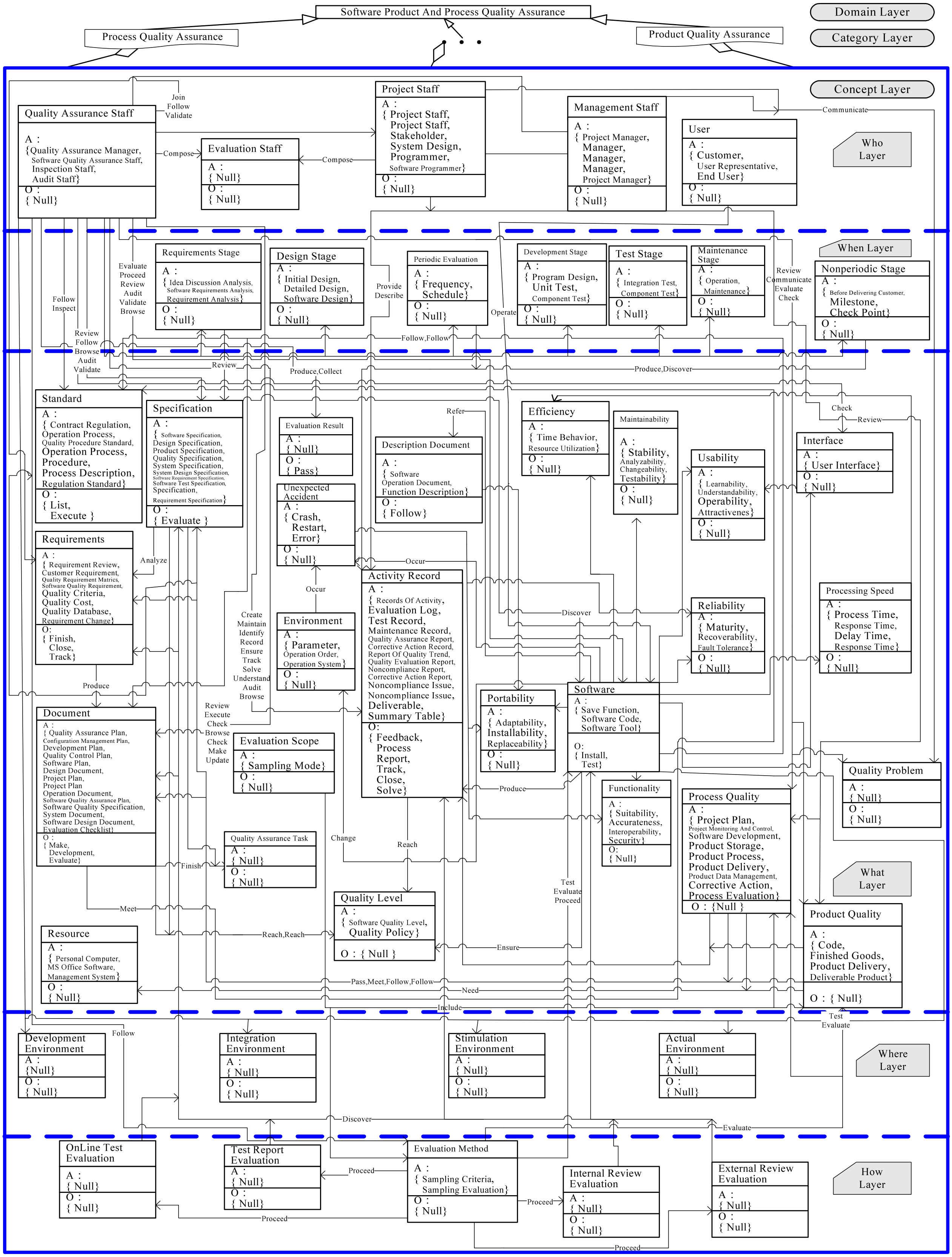

Software Product And Process Quality Assurance
Process Quality Assurance
Product Quality Assurance
Domain Layer
Category Layer
Concept Layer
Who Layer
When Layer
What Layer
Where Layer
How Layer
Quality Assurance Staff
A : {Quality Assurance Manager, Software Quality Assurance Staff, Inspection Staff, Audit Staff}
O : { Null}
Join Follow Validate
Compose
Evaluation Staff
A : { Null}
O : { Null}
Project Staff
A : { Project Staff, Project Staff, Stakeholder, System Design, Programmer, Software Programmer}
O : { Null}
Management Staff
A : { Project Manager, Manager, Manager, Manager, Project Manager}
O : { Null}
User
A : { Customer, User Representative, End User}
O : { Null}
Communicate
Evaluate Proceed Review Audit Validate Browse
Follow Inspect
Review Follow Browse Audit Validate
Requirements Stage
A : { Idea Discussion Analysis, Software Requirements Analysis, Requirement Analysis}
O : { Null}
Design Stage
A : { Initial Design, Detailed Design, Software Design}
O : { Null}
Provide Describe
Periodic Evaluation
A : { Frequency, Schedule}
O : { Null}
Operate
Development Stage
A : { Program Design, Unit Test, Component Test}
O : { Null}
Test Stage
A : { Integration Test, Component Test}
O : { Null}
Maintenance Stage
A : { Operation, Maintenance}
O : { Null}
Review Communicate Evaluate Check
Nonperiodic Stage
A : { Before Delivering Customer, Milestone, Check Point}
O : { Null}
Follow,Follow
Review
Produce,Collect
Produce,Discover
Refer
Check
Review
Standard
A : { Contract Regulation, Operation Process, Quality Procedure Standard, Operation Process, Procedure, Process Description, Regulation Standard}
O : { List, Execute }
Specification
A : { Software Specification, Design Specification, Product Specification, Quality Specification, System Specification, System Design Specification, Software Requirement Specification, Software Test Specification, Specification, Requirement Specification}
O : { Evaluate }
Evaluation Result
A : { Null}
O : { Pass}
Description Document
A : { Software Operation Document, Function Description}
O : { Follow}
Efficiency
A : { Time Behavior, Resource Utilization}
O : { Null}
Maintainability
A : { Stability, Analyzability, Changeability, Testability}
O : { Null}
Usability
A : { Learnability, Understandability, Operability, Attractivenes}
O : { Null}
Interface
A : { User Interface}
O : { Null}
Unexpected Accident
A : { Crash, Restart, Error}
O : { Null}
Occur
Requirements
A : { Requirement Review, Customer Requirement, Quality Requirement Matrics, Software Quality Requirement, Quality Criteria, Quality Cost, Quality Database, Requirement Change}
O: { Finish, Close, Track}
Analyze
Activity Record
A : { Records Of Activity, Evaluation Log, Test Record, Maintenance Record, Quality Assurance Report, Corrective Action Record, Report Of Quality Trend, Quality Evaluation Report, Noncompliance Report, Corrective Action Report, Noncompliance Issue, Noncompliance Issue, Deliverable, Summary Table}
O: { Feedback, Process Report, Track, Close, Solve}
Occur
Environment
A : { Parameter, Operation Order, Operation System}
O : { Null}
Create Maintain Identify Record Ensure Track Solve Understand Audit Browse
Discover
Reliability
A : { Maturity, Recoverability, Fault Tolerance}
O : { Null}
Processing Speed
A : { Process Time, Response Time, Delay Time, Response Time}
O : { Null}
Produce
Document
A : { Quality Assurance Plan, Configuration Management Plan, Development Plan, Quality Control Plan, Software Plan, Design Document, Project Plan, Project Plan Operation Document, Software Quality Assurance Plan, Software Quality Specification, System Document, Software Design Document, Evaluation Checklist}
O : { Make, Development, Evaluate}
Review Execute Check Browse Check Make Update
Evaluation Scope
A : { Sampling Mode}
O : { Null}
Portability
A : { Adaptability, Installability, Replaceability}
O : { Null}
Software
A : { Save Function, Software Code, Software Tool}
O: { Install, Test}
Produce
Quality Problem
A : { Null}
O : { Null}
Functionality
A : { Suitability, Accurateness, Interoperability, Security}
O: { Null}
Process Quality
A : { Project Plan, Project Monitoring And Control, Software Development, Product Storage, Product Process, Product Delivery, Product Data Management, Corrective Action, Process Evaluation}
O : {Null }
Quality Assurance Task
A : { Null}
O : { Null}
Finish
Change
Reach
Test Evaluate Proceed
Quality Level
A : { Software Quality Level, Quality Policy}
O : { Null }
Meet
Product Quality
A : { Code, Finished Goods, Product Delivery, Deliverable Product}
O : { Null}
Resource
A : { Personal Computer, MS Office Software, Management System}
O : { Null}
Reach,Reach
Ensure
Pass,Meet,Follow,Follow
Need
Include
Test Evaluate
Follow
Development Environment
A : {Null}
O : { Null}
Integration Environment
A : { Null}
O : { Null}
Stimulation Environment
A : { Null}
O : { Null}
Actual Environment
A : { Null}
O : { Null}
Discover
Evaluate
OnLine Test Evaluation
A : { Null}
O : { Null}
Test Report Evaluation
A : { Null}
O : { Null}
Proceed
Evaluation Method
A : { Sampling Criteria, Sampling Evaluation}
O : { Null}
Proceed
Internal Review Evaluation
A : { Null}
O : { Null}
External Review Evaluation
A : { Null}
O : { Null}
Proceed
Proceed

Fig. 4 **a** Structure of the partial PPQA ontology. **b** Corresponding Chinese version of (**a**)

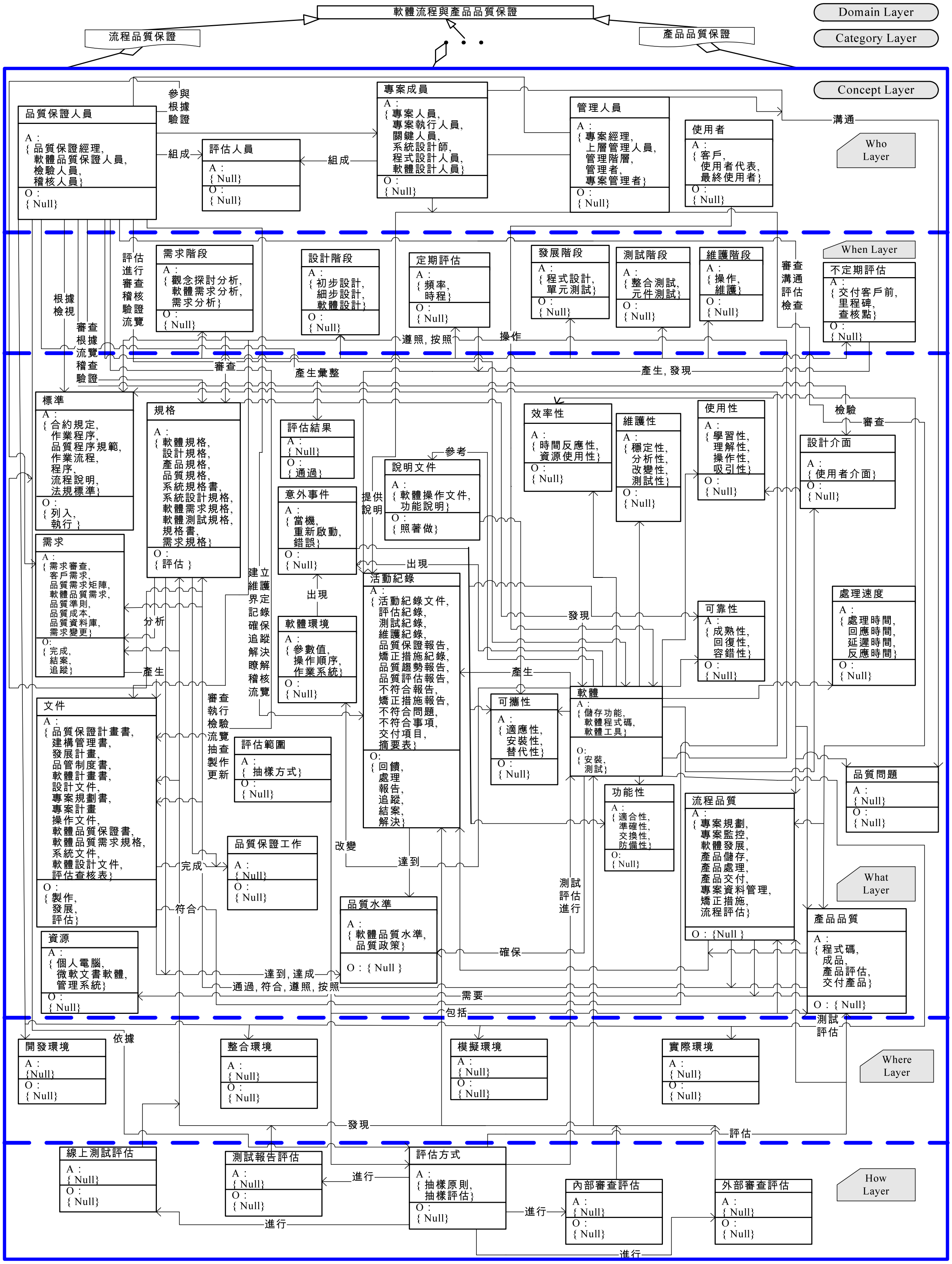

軟體流程與產品品質保證
流程品質保證
產品品質保證
Domain Layer
Category Layer
Concept Layer
Who Layer
When Layer
What Layer
Where Layer
How Layer
品質保證人員
A : { 品質保證經理, 軟體品質保證人員, 檢驗人員, 稽核人員 }
O : { Null}
參與 根據 驗證
組成
評估人員
A : { Null}
O : { Null}
專案成員
A : { 專案人員, 專案執行人員, 關鍵人員, 系統設計師, 程式設計人員, 軟體設計人員 }
O : { Null}
管理人員
A : { 專案經理, 上層管理人員, 管理階層, 管理者, 專案管理者 }
O : { Null}
使用者
A : { 客戶, 使用者代表, 最終使用者 }
O : { Null}
溝通
需求階段
A : { 觀念探討分析, 軟體需求分析, 需求分析 }
O : { Null}
設計階段
A : { 初步設計, 細步設計, 軟體設計 }
O : { Null}
定期評估
A : { 頻率, 時程 }
O : { Null}
發展階段
A : { 程式設計, 單元測試 }
O : { Null}
測試階段
A : { 整合測試, 元件測試 }
O : { Null}
維護階段
A : { 操作, 維護 }
O : { Null}
不定期評估
A : { 交付客戶前, 里程碑, 查核點 }
O : { Null}
評估 進行 審查 稽核 驗證 流覽
根據 檢視
審查 根據 流覽 稽查 驗證
審查 溝通 評估 檢查
遵照, 按照
操作
審查
產生彙整
產生, 發現
檢驗
審查
標準
A : { 合約規定, 作業程序, 品質程序規範, 作業流程, 程序, 流程說明, 法規標準 }
O : { 列入, 執行 }
規格
A : { 軟體規格, 設計規格, 產品規格, 品質規格, 系統規格書, 系統設計規格, 軟體需求規格, 軟體測試規格, 規格書, 需求規格 }
O : { 評估 }
評估結果
A : { Null}
O : { 通過 }
參考
說明文件
A : { 軟體操作文件, 功能說明 }
O : { 照著做 }
提供 說明
意外事件
A : { 當機, 重新啟動, 錯誤 }
O : { Null}
效率性
A : { 時間反應性, 資源使用性 }
O : { Null}
維護性
A : { 穩定性, 分析性, 改變性, 測試性 }
O : { Null}
使用性
A : { 學習性, 理解性, 操作性, 吸引性 }
O : { Null}
設計介面
A : { 使用者介面 }
O : { Null}
需求
A : { 需求審查, 客戶需求, 品質需求矩陣, 軟體品質需求, 品質準則, 品質成本, 品質資料庫, 需求變更 }
O: { 完成, 結案, 追蹤 }
分析
產生
建立 維護 界定 記錄 確保 追蹤 解決 瞭解 稽核 流覽
出現
出現
活動紀錄
A : { 活動紀錄文件, 評估紀錄, 測試紀錄, 維護紀錄, 品質保證報告, 矯正措施紀錄, 品質趨勢報告, 品質評估報告, 不符合報告, 矯正措施報告, 不符合問題, 不符合事項, 交付項目, 摘要表 }
O: { 回饋, 處理, 報告, 追蹤, 結案, 解決 }
軟體環境
A : { 參數值, 操作順序, 作業系統 }
O : { Null}
發現
可靠性
A : { 成熟性, 回復性, 容錯性 }
O : { Null}
處理速度
A : { 處理時間, 回應時間, 延遲時間, 反應時間 }
O : { Null}
文件
A : { 品質保證計畫書, 建構管理書, 發展計畫, 品管制度書, 軟體計畫書, 設計文件, 專案規劃書, 專案計畫, 操作文件, 軟體品質保證書, 軟體品質需求規格, 系統文件, 軟體設計文件, 評估查核表 }
O : { 製作, 發展, 評估 }
審查 執行 檢驗 流覽 抽查 製作 更新
評估範圍
A : { 抽樣方式 }
O : { Null}
產生
可攜性
A : { 適應性, 安裝性, 替代性 }
O : { Null}
軟體
A : { 儲存功能, 軟體程式碼, 軟體工具 }
O: { 安裝, 測試 }
品質問題
A : { Null}
O : { Null}
功能性
A : { 適合性, 準確性, 交換性, 防備性 }
O: { Null}
流程品質
A : { 專案規劃, 專案監控, 軟體發展, 產品儲存, 產品處理, 產品交付, 專案資料管理, 矯正措施, 流程評估 }
O : { Null }
品質保證工作
A : { Null}
O : { Null}
完成
改變
達到
測試 評估 進行
品質水準
A : { 軟體品質水準, 品質政策 }
O : { Null }
符合
確保
產品品質
A : { 程式碼, 成品, 產品評估, 交付產品 }
O : { Null}
資源
A : { 個人電腦, 微軟文書軟體, 管理系統 }
O : { Null}
達到, 達成
通過, 符合, 遵照, 按照
需要
包括
測試 評估
依據
開發環境
A : {Null}
O : { Null}
整合環境
A : { Null}
O : { Null}
模擬環境
A : { Null}
O : { Null}
實際環境
A : { Null}
O : { Null}
發現
評估
線上測試評估
A : { Null}
O : { Null}
測試報告評估
A : { Null}
O : { Null}
進行
評估方式
A : { 抽樣原則, 抽樣評估 }
O : { Null}
進行
內部審查評估
A : { Null}
O : { Null}
進行
外部審查評估
A : { Null}
O : { Null}
進行
進行

Fig. 4 (*continued*)

of this ontology is "Software Process and Product Quality Assurance". In the *category layer*, the "Process Quality Assurance" and "Product Quality Assurance" are two categories of the ontology. Every concept in the *concept layer* contains a concept name, an attribute set, and an operation set. For instance, the *who layer* contains three concepts, namely "Quality Assurance Staff", "Project Staff", and "Management Staff". The attribute set of the concept "Quality Assurance Staff" is {Quality Assurance Manager, Software Quality Assurance Staff, Inspection Staff, Audit Staff}. As another example, the *what layer* contains a concept called "Software Description Document". Its attribute set is {Software Operation Description, Function Description}, and its operation set is {Follow}. With the support of the ontology, the ontology-based computational intelligent multi-agent would be able to extract the key sentences from the evaluated reports to make the relevant stakeholders understand quickly and easily the level of adherence to the applicable process descriptions, standards, and procedures.

5 Computational intelligent multi-agent for CMMI assessment

The computational intelligent multi-agent comprises three modules, namely the natural language processing agent, ontological reasoning agent, and summary agent, which are described below.

5.1 Natural language processing agent

The natural language processing agent includes a *Part-of-Speech* (*POS*) tagger and a term filter. Its main task is to tag the terms of evaluation reports, then to filter meaningless terms to preserve the terms with a noun or verb as *POS*, and finally to pass the results to the ontological reasoning agent. Three factors, namely the *POS* similarity, the *Term Word* (*TW*) similarity, and the *Semantic Distance* (*SD*) similarity, were chosen as the conceptual similarity factors for analyzing the Chinese terms and determining the conceptual similarity between any two Chinese terms based on the features of the Chinese language and the definitions of the Chinese Dictionary.

The *POS* similarity is the path length between two nodes located on the tagging tree [1]. The tagging tree is utilized to determine the conceptual similarity in *POS* between any two Chinese terms. For example, if a term pair (Quality Assurance Staff, Join) with the *POS* (Na, VC) exists, then the *POS* similarity of this term pair is 4 (Na→N→NV→V→VC), where Na, N, NV, V, and VC represent common noun, noun, noun-verb, verb and transitive verb, respectively. The *TW* similarity denotes the value of the conceptual similarity between any two Chinese terms based on three criteria [18]: (1) a pair containing more identical words in both terms indicates that the terms are more similar to each other in semantic meaning; (2) terms in a pair with both identical and continuous words have much greater semantic similarity than those in a pair without identical or continuous words, and (3) terms in a pair with identical starting or ending words have a strong semantic similarity. For example, the term pair (Requirement Stage, Design Stage), whose Chinese term pair is (需求階段, 設計階段), has two identical Chinese words, "階" and "段", and the identical ending Chinese word, "段", so the value of *TW* similarity of this term pair is 2.5, i.e., $2+0.5=2.5$. The *SD* similarity indicates the semantic distance in the same layer of the concept layer for any term pair. A closer distance indicates that terms are more similar to each other in semantic meaning. For instance, consider the term pair (Quality Assurance Staff, Project Staff). These two terms are both located in the *who layer* of the constructed PPQA ontology. Hence, the *SD* similarity between these two terms is 1, i.e., Quality Assurance Staff → Project Staff.

5.2 Ontological reasoning agent

The ontological reasoning agent adopts the results of the natural language processing agent of the evaluation report and the concepts of the PPQA ontology to reason the term relation strength. The algorithm of the ontological reasoning agent appears below. Table 1 shows the fuzzy rules predefined by domain experts.

A trapezoidal membership function for fuzzy set *FS* specified by four parameters $FS(x:a,b,c,d)$ is given in (1). Figure 5(a) shows the trapezoidal membership functions for fuzzy sets $POS\_Low(x:4,5.2,6,6)$, $POS\_Medium(x:1,2.5,3.5,5)$, and $POS\_High(x:0,0,0.8,2)$. Figure 5(b) shows the trapezoidal membership functions for fuzzy sets $TW\_Low(x:0,\ 0,\ 0.12,\ 0.3)$, $TW\_Medium(x:0.2,\ 0.43,\ 0.58,\ 0.8)$, and $TW\_High(x:0.7,\ 0.88,\ 1,\ 1)$. Figure 5(c) displays the trapezoidal membership functions for fuzzy sets $SD\_Low(x:17,\ 21.8,\ 25,\ 25)$, $SD\_Medium(x:4,\ 10,\ 14,\ 20)$, and $SD\_High(x:0,\ 0,\ 2.8,\ 7)$. Figure 5(d) displays the trapezoidal membership functions for fuzzy set $TRS\_VeryLow(x:0,\ 0,\ 0.12,\ 0.3)$, $TRS\_Low(x:0.2,\ 0.31,\ 0.39,\ 0.5)$, $TRS\_Medium(x:0.3,\ 0.45,\ 0.55,\ 0.7)$,

Ontological Reasoning Agent Algorithm

```
BEGIN
  Input the ontological concept set $C_O = \{C_{O_1}, C_{O_2}, ..., C_{O_M}\}$ for the domain ontology $DO$.
  Input the term set $T_{NLP} = \{T_{NLP_1}, T_{NLP_2}, ..., T_{TNL_N}\}$ for the evaluation report $ER$.
  Input fuzzy set $A_{in} = \{POS\_Low, POS\_Medium, POS\_High, TW\_Low, TW\_Medium, TW\_High, SD\_Low, SD\_Medium, SD\_High\}$
  and output fuzzy set $A_{out} = \{TRS\_VeryLow, TRS\_Low, TRS\_Medium, TRS\_High, TRS\_VeryHigh\}$
  Input fuzzy inference rules set $FIR = \{Rule\ 1, Rule\ 2, ..., Rule\ K\}$
  Initialize $i = 1$
  DO UNTIL ($i > M$)
     Initialize $j = 1$
     DO UNTIL ($j > N$)
        Let $V_{ij} = (C_{O_i}, T_{NLP_j})$ be the term pair.
          $X_{ij} = (V_{ij-POS}, V_{ij-TW}, V_{ij-SD})$, where $V_{ij-POS}$ is the part-of-speech path length of each term pair, $V_{ij-TW}$ is the number of
          the same words of each term pair, and $V_{ij-SD}$ is the semantic distance of each term pair.
        Initialize $k = 1$
        DO UNTIL ($k > K$)
           Calculate the matching degree of the Rule $k$ by $\mu_{ij\_k} = MIN(\mu_{A_{in}}(X_{ij}))$
           Calculate the center of area of the Rule $k$ by $y_{A_{out}ij\_k} = COA(\mu_{ij\_k})$
           increment $k$
        END DO UNTIL
        Initialize $p = 1$
        DO UNTIL ($p > P$)
           Calculate the membership values of $X_{ij}$ to the fuzzy classes $F_{ij\_p}$ by
              $y_{ij\_p} = MAX(y_{A_{out}ij\_k})$
           where $F_{ij\_p}$ means the fuzzy class for all of fuzzy rules. Each fuzzy class is an aggregation of the fired rules that have the same
           consequences.
           increment $p$
        END DO UNTIL
        $TRS_{ij} = \sum_{p=1}^{P} w_{ij\_p} y_{ij\_p} \Big/ \sum_{p=1}^{P} w_{ij\_p}$, where $w_{ij\_p}$ means the weight for $y_{ij\_p}$ and $P$ means the number of linguistic terms of the
        output fuzzy variable, $TRS$.
        increment $j$
     END DO UNTIL
     increment $i$
  END DO UNTIL
END
```

TRS_High(x : 0.5, 0.61, 0.69, 0.8), and *TRS_VeryHigh*(x : 0.7, 0.88, 1, 1).

$$FS(x : a, b, c, d) = \begin{cases} 0 & x < a\,, \\ (x-a)/(b-a) & a \le x < b, \\ 1 & b \le x \le c, \\ (d-x)/(d-c) & c < x \le d, \\ 0 & x > d. \end{cases} \tag{1}$$

5.3 Summary agent

The summary agent has five sub-processes, namely the matched concept finder, the sentence path extractor, the sentence path filter, the sentence generator, and the compression rate computation. Based on the inferred *TRS* results of the ontological reasoning agent and the pre-constructed CMMI domain ontology, the matched concept finder first selects the term pairs whose *TRS* values are over the threshold σ_{TRS}, and then finds the matched concepts. The sentence path extractor then exploits the matched concepts along with the pre-constructed CMMI ontology to search all of the possible sentence paths by following the *Depth-First-Search* algorithm. The sentence path filter then removes the redundant sentence paths. The aim of the sentence generator is to produce the key sentences of the evaluation report according to the retained sentence paths. Finally, the compression rate computation derives the compression rate of the evaluation report. The summary agent algorithm of each process is given below.

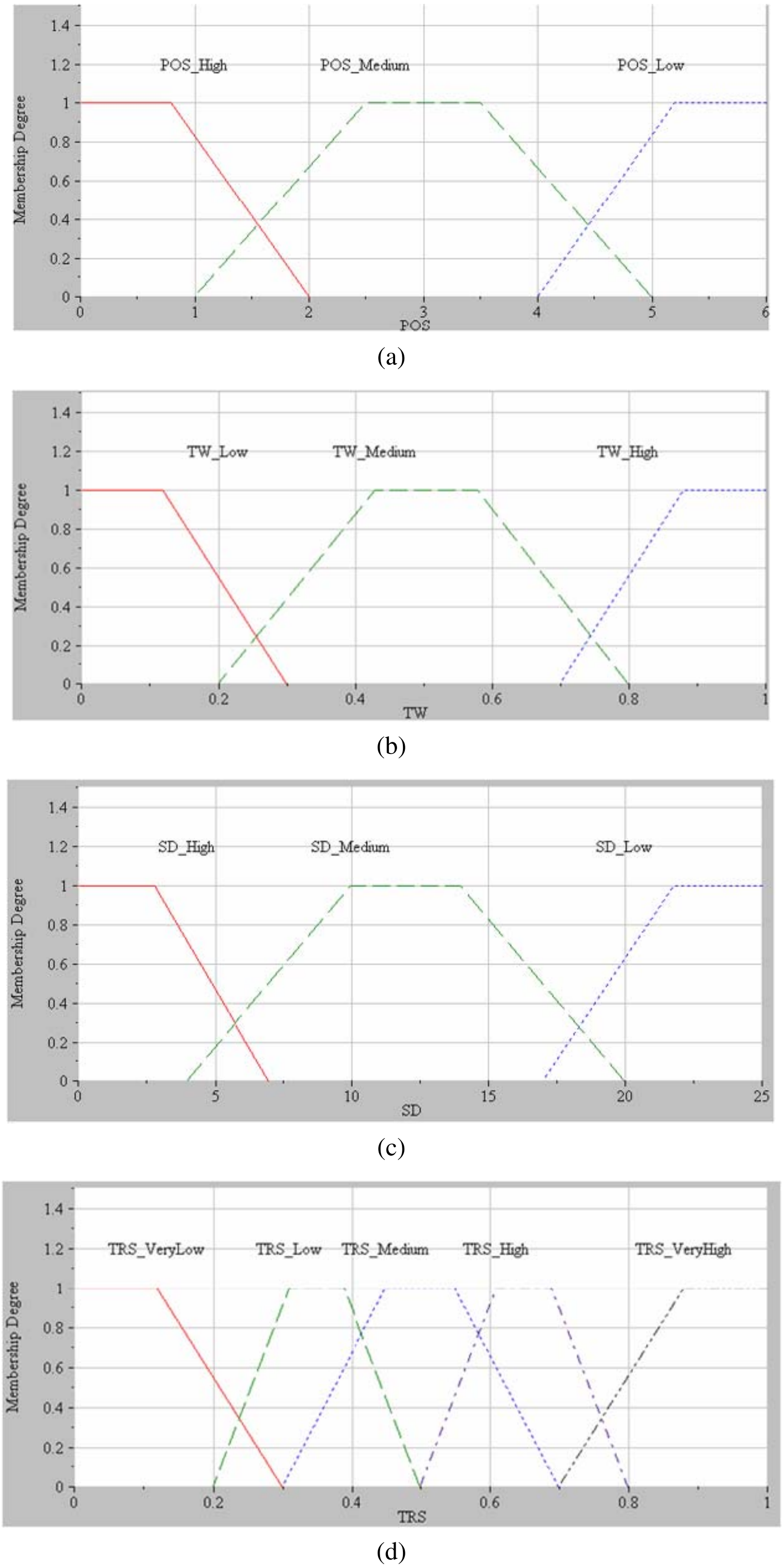
Table 1 Fuzzy rules of the ontological reasoning agent

Rule 1: if *POS* is **Low** and *TW* is **Low** and *SD* is **Low** then *TRS* is **Very Low**
Rule 2: if *POS* is **Low** and *TW* is **Low** and *SD* is **Medium** then *TRS* is **Low**
Rule 3: if *POS* is **Low** and *TW* is **Low** and *SD* is **High** then *TRS* is **Low**
Rule 4: if *POS* is **Low** and *TW* is **Medium** and *SD* is **Low** then *TRS* is **Low**
Rule 5: if *POS* is **Low** and *TW* is **Medium** and *SD* is **Medium** then *TRS* is **Low**
Rule 6: if *POS* is **Low** and *TW* is **Medium** and *SD* is **High** then *TRS* is **Medium**
Rule 7: if *POS* is **Low** and *TW* is **High** and *SD* is **Low** then *TRS* is **Low**
Rule 8: if *POS* is **Low** and *TW* is **High** and *SD* is **Medium** then *TRS* is **Medium**
Rule 9: if *POS* is **Low** and *TW* is **High** and *SD* is **High** then *TRS* is **High**
Rule 10: if *POS* is **Medium** and *TW* is **Low** and *SD* is **Low** then *TRS* is **Low**
Rule 11: if *POS* is **Medium** and *TW* is **Low** and *SD* is **Medium** then *TRS* is **Low**
Rule 12: if *POS* is **Medium** and *TW* is **Low** and *SD* is **High** then *TRS* is **Medium**
Rule 13: if *POS* is **Medium** and *TW* is **Medium** and *SD* is **Low** then *TRS* is **Low**
Rule 14: if *POS* is **Medium** and *TW* is **Medium** and *SD* is **Medium** then *TRS* is **Medium**
Rule 15: if *POS* is **Medium** and *TW* is **Medium** and *SD* is **High** then *TRS* is **High**
Rule 16: if *POS* is **Medium** and *TW* is **High** and *SD* is **Low** then *TRS* is **Medium**
Rule 17: if *POS* is **Medium** and *TW* is **High** and *SD* is **Medium** then *TRS* is **High**
Rule 18: if *POS* is **Medium** and *TW* is **High** and *SD* is **High** then *TRS* is **High**
Rule 19: if *POS* is **High** and *TW* is **Low** and *SD* is **Low** then *TRS* is **Low**
Rule 20: if *POS* is **High** and *TW* is **Low** and *SD* is **Medium** then *TRS* is **Medium**
Rule 21: if *POS* is **High** and *TW* is **Low** and *SD* is **High** then *TRS* is **High**
Rule 22: if *POS* is **High** and *TW* is **Medium** and *SD* is **Low** then *TRS* is **Medium**
Rule 23: if *POS* is **High** and *TW* is **Medium** and *SD* is **Medium** then *TRS* is **High**
Rule 24: if *POS* is **High** and *TW* is **Medium** and *SD* is **High** then *TRS* is **High**
Rule 25: if *POS* is **High** and *TW* is **High** and *SD* is **Low** then *TRS* is **High**
Rule 26: if *POS* is **High** and *TW* is **High** and *SD* is **Medium** then *TRS* is **High**
Rule 27: if *POS* is **High** and *TW* is **High** and *SD* is **High** then *TRS* is **Very High**

Matched Concept Finder Algorithm

```
BEGIN
  Input term relation strength set $TRS = \{TRS_{11}, TRS_{12}, ..., TRS_{MN}\}$ for the evaluation report ER.
  Input the matched term set $T_{mNLP} = \{T_{mNLP_1}, T_{mNLP_2}, ..., T_{mNLP_N}\}$ for the evaluation report ER.
  Input the domain ontology DO.
  Input the ontological concept set $C_O$.
  Initialize the matched ontological concepts $C_{mO} = \phi$.
  Initialize $i = 1$
  DO UNTIL ($i > M$)
     Initialize $j = 1$
     DO UNTIL ($j > N$)
        IF ( $TRS_{ij} \geq \sigma_{TRS}$ ) THEN /* where $\sigma_{TRS}$ denotes the membership degree threshold for the TRS value*/
           Find out the matched concept $C_{mO_i}$ according to the $T_{mNLP_j}$, $C_O$, and DO.
           Find out the attributes set $\{A_{C_{mOi}}1, ..., A_{C_{mOi}}q_j\}$ and operation set $\{O_{C_{mOi}}1, ..., O_{C_{mOi}}q_j\}$ of the concept $C_{mO_i}$.
           Add the matched concept $C_{mO_i}$ to the $C_{mO}$ set.
        END IF
        increment j
     END DO UNTIL
     increment i
  END DO UNTIL
END
```

Fig. 5 Trapezoidal membership functions for fuzzy variables (**a**) *POS*, (**b**) *TW*, (**c**) *SD*, and (**d**) *TRS*



(a)

(b)

(c)

(d)

Sentence Path Extractor Algorithm

```
BEGIN
  Input the matched ontological concepts $C_{mO} = \{C_{mO_1}, C_{mO_2}, ..., C_{mO_M}\}$.
  Input the domain ontology DO.
  Initialize the extracted sentence path set ESP set = $\phi$.
  Initialize AdjMatrix( $M \times M$ ) = 0 and $i$ = 1
  DO UNTIL ($i$ > M)
    IF there is a relation between $C_{mO_i}$ and $C_{mO_{(i+1)}}$ for the domain ontology DO THEN
      AdjMatrix($i$) = 1
    END IF
    increment $i$
  END DO UNTIL
  Initialize $i$ = 1
  DO UNTIL ($i > M$)
    Compute the out-degree value of the $C_{mO_i}$ concept according to the AdjMatrix($i$).
    IF (out-degree != 0) THEN
      According to the adjacency matrix AdjMatrix, execute the Depth-First Search Algorithm to search the possible sentence paths.
      Generate the extracted sentence paths in the format of [Concept Name] → [Concept Name] → … → [Concept Name].
      Add the extracted sentence path $ESP_i$ to the ESP set.
    END IF
    increment $i$
  END DO UNTIL
END
```

Sentence Path Filter Algorithm

```
BEGIN
  Input the extracted sentence path set $ESP = \{ESP_1, ESP_2, ..., ESP_N\}$
  Initialize a retained sentence path set $RSP$ set = $\phi$.
  Initialize $i = 2$
  DO UNTIL ($i > N$)
    Initialize $j = 1$
    DO UNTIL ($j > N - 1$)
      IF ( $Length(ESP_j) < Length(ESP_{j+1})$ ) THEN
        SWAP( $ESP_j$ , $ESP_{j+1}$ )
        where $Length(ESP_j)$ means the length of extracted sentence path $ESP_j$
      END IF
      increment $j$
    END DO UNTIL
    increment $i$
  END DO UNTIL
  Initialize $i = 1$
  DO UNTIL ($i > N$)
    Retrieve the concept information of extracted sentence path $ESP_i$.
    Initialize $j = i + 1$
    DO UNTIL ($j$ > N)
      Retrieve the concept information of extracted sentence path $ESP_j$.
      IF $ESP_i \subseteq ESP_j$ THEN
        Label $ESP_j$ as filtered state.
      END IF
      increment $j$
    END DO UNTIL
    increment $i$
  END DO UNTIL
  Initialize $i = 1$
  DO UNTIL ($i > N$)
    IF $ESP_i$ is not labeled as filtered state THEN
      Rename $ESP_i$ to $RSP_i$ and add it to the RSP set.
    END IF
    increment $i$
  END DO UNTIL
END
```

Sentence Generator Algorithm

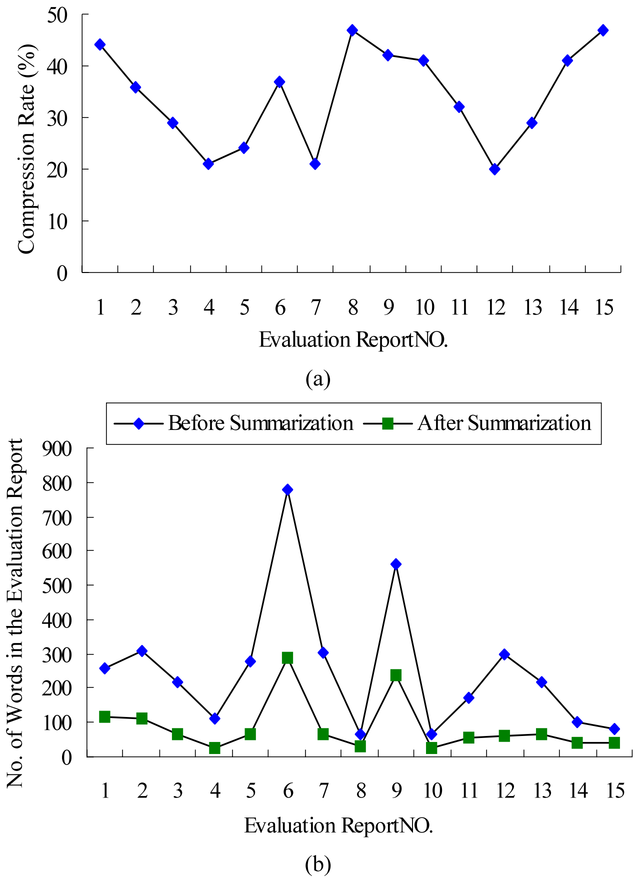
```
BEGIN
  Input a retained sentence path set  $RSP = \{RSP_1, RSP_2, ..., RSP_N\}$
  Input the matched ontological concepts  $C_{mO} = \{C_{mO_1}, C_{mO_2}, ..., C_{mO_M}\}$
  Initialize a sentence generator set GS set = $\phi$ .
  Initialize $i = 1$
  DO UNTIL ($i > N$)
     Initialize $j = 1$
     DO UNTIL ($j > r_j$)
        Retrieve the attribute set and operation set of the matched ontological concepts $C_{mO_j}$ and $C_{mo_{(j+1)}}$ , for the retained sentence path $RSP_i$ .
        Get the relation $R_{C_{mO_j} C_{mO_{(j+1)}}}$ between $C_{mO_j}$ and $C_{mO_{(j+1)}}$ .
        Generate the sentence in the format of [Concept Name, Attribute, Operation] $\underrightarrow{Relation}$ [Concept Name, Attribute, Operation] $\underrightarrow{Relation}$ … $\underrightarrow{Relation}$ [Concept Name, Attribute, Operation].
        where $r_j$ denotes the number of matched ontological concepts for $RSP_i$
        increment $j$
     END DO UNTIL
     Add the generated sentence $GS_i$ to the GS set.
     increment $i$
  END DO UNTIL
END
```

Compression Rate Computation Algorithm

```
BEGIN
  Input an Evaluation Report  $ER_N$ .
  Input the generated sentence set  $GS_N = \{GS_1, GS_2, ..., GS_M\}$  for the evaluation report  $ER_N$ .
     Compute the number of words in the GS by
        $NOWords_{after} = \sum_{i=1}^{M} No.\,of\ words\ in\ the\ GS_i$ .
     Compute the number of words in the $ER_N$ by
        $NOWords_{before} = No.\,of\ words\ in\ the\ ER_N$
     $Compression\ Rate = \frac{NOWords_{after}}{NOWords_{before}} \times 100\%$
END
```

Table 2 Compression rate and the number of words before and after summarization of each evaluation report

Evaluation report	No. of words in the evaluation report		Compression rate
NO	Before summarization	After summarization	(%)
1	258	115	44
2	307	113	36
3	217	64	29
4	110	24	21
5	278	67	24
6	777	288	37
7	302	65	21
8	65	31	47
9	559	236	42
10	65	27	41
11	173	57	32
12	296	60	20
13	217	64	29
14	100	41	41
15	80	38	47



(a)

(b)

Fig. 6 **a** The curve of the compression rate for the tested fifteen evaluation reports. **b** The curves of the number of words in each evaluation report before and after summarization

6 Experimental results

The proposed ontology-based computational intelligent multi-agent for CMMI assessment was implemented in the Java programming language. The experimental environment was built to test the performance of the proposed approach. The experimental Chinese PPQA evaluation reports were retrieved from project members involved in a CMMI project at the National University of Tainan.

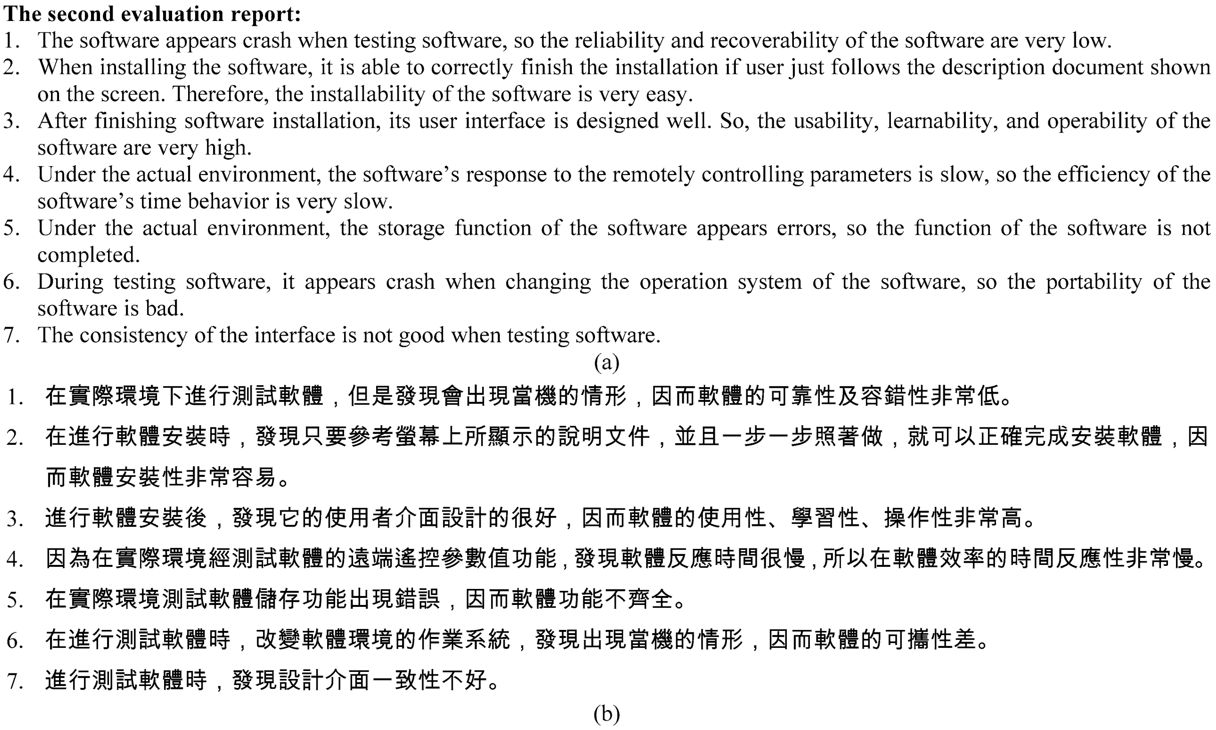
The first experiment was to measure the performance of the ontology-based computational intelligent multi-agent system from the compression rate. Figure 6(a) shows the curve of the compression rate for the fifteen evaluation reports. The second experiment was to observe the performance of the proposed approach from the variance in the number of words in the evaluation report before and after summarization. Figure 6(b) shows the curve of the number of words in each evaluation report before and after summarization. Figure 6 reveals that the compression rate for these fifteen evaluation reports ranged between 20% and 50%, and that the number of words decreased after summarization. Therefore, the ontology-based computational intelligent multi-agent can effectively summarize the evaluation reports. Table 2 lists the values of the compression rate and the number of words before and after summarization of each evaluation report.

Finally, the 2nd and the 13th evaluation reports were analyzed. Figures 7, 8, and 9 show the experimental results of the second evaluation report in English and Chinese. According to Table 2, the numbers of words in the 2nd evaluation report before and after summarization were 307 and 113, respectively. Consequently, the compression rate for the 2nd evaluation report was 36%. Similarly, Figs. 10, 11, and 12 show the experimental results of the 13th evaluation report in English and Chinese, where the number of words before and after summarization were 217 and 64, respectively and the compression rate was 29%.

7 Conclusions

This study presents an ontology-based computational intelligent mutli-agent for CMMI assessment. The proposed computational intelligent multi-agent comprises a natural language processing agent, an ontological reasoning agent, and a summary agent. Experimental results indicate that the ontology-based computational intelligent multi-agent can effectively summarize the evaluation reports. Additionally, some problems still require further study. For example, the number of tested evaluation reports should be raised to enhance the performance of the ontology-based computational intelligent multi-agent, and the ontology should be improved to extract necessary and key sentences. Moreover, the summary may contain sentences that are not strongly related. This problem is also one of the common bottlenecks in many current summary generation systems. For instance, the reliability of the software plays a significant role in software quality, but the current proposed approach cannot extract the “reliability is low” from the second evaluation report. Additionally, the proposed approach will be extended to summarize the final evaluation results after matching with the applicable process descriptions, standards and procedures, so that the relevant stakeholders can quickly understand the adherence of the project’s process and product quality.

Acknowledgements The authors would like to thank the anonymous reviewers for their constructive and useful comments, and also would like to thank the support sponsored by the Department of Industrial Technology, Ministry of Economic Affairs, R.O.C. under the grant no. 96-EC-17-A-02-S1-029, and by the National Science Council of Taiwan under the grant no. NSC 95-2221-E-024-009-MY2.



The second evaluation report:

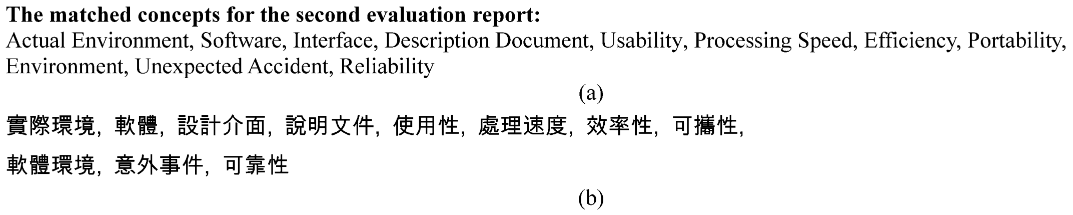
1. The software appears crash when testing software, so the reliability and recoverability of the software are very low.
2. When installing the software, it is able to correctly finish the installation if user just follows the description document shown on the screen. Therefore, the installability of the software is very easy.
3. After finishing software installation, its user interface is designed well. So, the usability, learnability, and operability of the software are very high.
4. Under the actual environment, the software's response to the remotely controlling parameters is slow, so the efficiency of the software's time behavior is very slow.
5. Under the actual environment, the storage function of the software appears errors, so the function of the software is not completed.
6. During testing software, it appears crash when changing the operation system of the software, so the portability of the software is bad.
7. The consistency of the interface is not good when testing software.

(a)

1. 在實際環境下進行測試軟體，但是發現會出現當機的情形，因而軟體的可靠性及容錯性非常低。
2. 在進行軟體安裝時，發現只要參考螢幕上所顯示的說明文件，並且一步一步照著做，就可以正確完成安裝軟體，因而軟體安裝性非常容易。
3. 進行軟體安裝後，發現它的使用者介面設計的很好，因而軟體的使用性、學習性、操作性非常高。
4. 因為在實際環境經測試軟體的遠端遙控參數值功能 , 發現軟體反應時間很慢 , 所以在軟體效率的時間反應性非常慢。
5. 在實際環境測試軟體儲存功能出現錯誤，因而軟體功能不齊全。
6. 在進行測試軟體時，改變軟體環境的作業系統，發現出現當機的情形，因而軟體的可攜性差。
7. 進行測試軟體時，發現設計介面一致性不好。

(b)

Fig. 7 The second evaluation report (**a**) in English, and (**b**) in Chinese



The matched concepts for the second evaluation report:

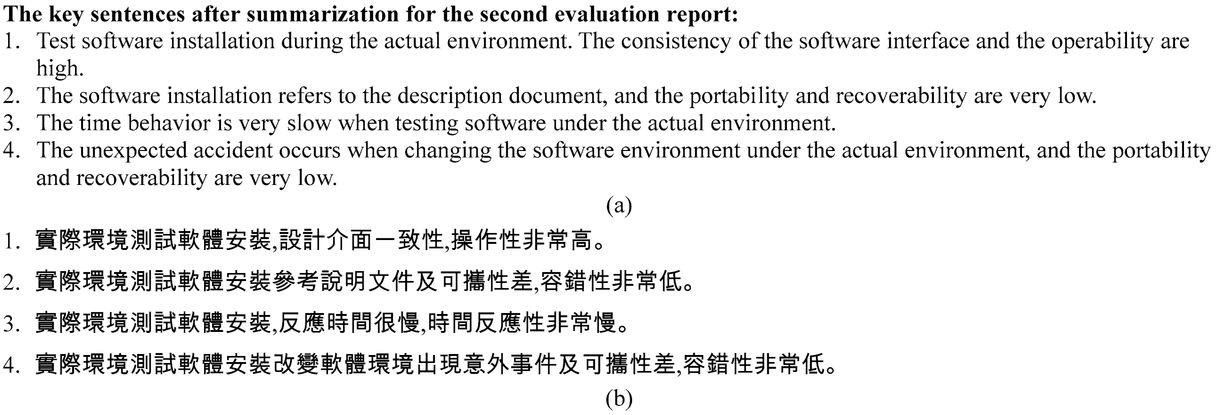
Actual Environment, Software, Interface, Description Document, Usability, Processing Speed, Efficiency, Portability, Environment, Unexpected Accident, Reliability

(a)

實際環境, 軟體, 設計介面, 說明文件, 使用性, 處理速度, 效率性, 可攜性, 軟體環境, 意外事件, 可靠性

(b)

Fig. 8 Matched concepts for the second evaluation report (**a**) in English, and (**b**) in Chinese



The key sentences after summarization for the second evaluation report:

1. Test software installation during the actual environment. The consistency of the software interface and the operability are high.
2. The software installation refers to the description document, and the portability and recoverability are very low.
3. The time behavior is very slow when testing software under the actual environment.
4. The unexpected accident occurs when changing the software environment under the actual environment, and the portability and recoverability are very low.

(a)

1. 實際環境測試軟體安裝,設計介面一致性,操作性非常高。
2. 實際環境測試軟體安裝參考說明文件及可攜性差,容錯性非常低。
3. 實際環境測試軟體安裝,反應時間很慢,時間反應性非常慢。
4. 實際環境測試軟體安裝改變軟體環境出現意外事件及可攜性差,容錯性非常低。

(b)

Fig. 9 The key sentences after summarization for the second evaluation report (**a**) in English, and (**b**) in Chinese

The thirteenth evaluation report:
The job of the software quality assurance during software development includes:
1. Join requirements review, and review the consistency between software specification and customer requirements in the requirement stage.
2. Review the software plan and configuration management plan in the requirement stage.
3. Execute the software plan, check the design interface, validate the quality requirement matrics and check the final design specification and operation documents in the design stage.
4. Review the software codes, and audit if the software codes meet the standardization in the development stage.
5. Audit the activity of records and review the product specifications in the test stage.

(a)

軟體開發階段之軟體品質保證工作內容:
1. 需求階段之觀念探討分析，參與需求審查，並審查軟體規格與客戶需求的一致性。
2. 需求階段之需求分析，要審查軟體計畫書及建構管理書。
3. 設計階段，要執行軟體計畫書、檢驗設計介面、驗證品質需求矩陣及檢驗最終設計規格及操作文件。
4. 發展階段，要審查軟體程式碼、稽核軟體程式碼是否標準化。
5. 測試階段，要稽核活動紀錄文件及審查產品規格。

(b)

Fig. 10 The thirteenth evaluation report (**a**) in English, and (**b**) in Chinese

The matched concepts for the thirteenth evaluation report:
Requirement Stage, Design Stage, Development Stage, Test Stage, Maintenance Stage, Activity of Record, Document Requirement, Specification, Software, Design Interface

(a)

需求階段，設計階段，發展階段，測試階段，維護階段，活動紀錄，文件
需求，規格，軟體，設計介面

(b)

Fig. 11 Matched concepts for the thirteenth evaluation report (**a**) in English, and (**b**) in Chinese

The key sentences after summarization for the thirteenth evaluation report:
1. Review specification and analyze requirement in the requirement stage and test stage.
2. Execute the documents in the test stage.
3. Check the design interface in the design stage.
4. Review and audit software and design interface in the development stage.
5. Validate the software and design interface in the maintenance stage.

(a)

1. 需求階段、測試階段審查規格分析需求。
2. 設計階段執行文件。
3. 設計階段檢驗設計介面。
4. 發展階段審查及稽核軟體及設計介面。
5. 維護階段驗證軟體及設計介面。

(b)

Fig. 12 (a) The main sentences after summarization for the thirteenth evaluation report (**a**) in English, and (**b**) in Chinese

References

1. Lee CS, Jian ZW, Huang LK (2005) A fuzzy ontology and its application to news summarization. IEEE Trans Syst Man Cybern Part B: Cybern 35(5):859–880
2. Ferber J (1999) Multi-agent systems. New York, Addison–Wesley
3. Chen H, Yen J (1996) Toward intelligent meeting agents. IEEE Comput 29(8):62–70
4. De len D, Pratt DB (2006) An integrated and intelligent DSS for manufacturing system. Expert Syst Appl 30(2):325–336
5. Soo VW, Lin SY, Yang SY, Lin SN, Cheng SL (2006) A cooperative multi-agent platform for invention based on patent document analysis and ontology. Expert Syst Appl 31(4):766–775
6. Hamdi MS (2006) MASACAD: a multiagent based approach to information customization. IEEE Intell Syst 21(1):60–67
7. Lee CS, Jiang CC, Hsieh TC (2006) A genetic agent using ontology model for meeting scheduling system. Inf Sci 176(9):1131–1155
8. Wang FY (2005) Agent-based control for networked traffic management systems. IEEE Intell Syst 20(5):92–96
9. Yan H, Jiang Y, Zheng J, Peng C, Li Q (2006) A multiplayer perceptron-based medical decision support system for heart disease diagnosis. Expert Syst Appl 30(3):272–281
10. Grossklags J, Schmidt C (2006) Software agents and market in efficiency: A human trader experiment. IEEE Trans Syst Man Cybern Part C: Appl Rev 36(1):1–13

11. Alani H, Kim S, Millard DE, Weal MJ, Hall W, Lewis PH, Shadbolt NR (2003) Automatic ontology-based knowledge extraction from web documents. IEEE Intell Syst 18(1):14–21
12. Chau KW (2007) An ontology-based knowledge management system for flow and water quality modeling. Adv Eng Softw 38(3):172–181
13. Corby O, Kuntz RD, Zucker CF, Gandon F (2006) Searching the semantic web: approximate query processing based on ontologies. IEEE Intell Syst 21(1):20–27
14. Navigli R, Velardi R, Gangemi A (2003) Ontology learning and its application to automated terminology translation. IEEE Intell Syst 18(1):22–31
15. Morbach J, Yang A, Marquardt W (2007) OntoCAPE-A large-scale ontology for chemical process engineering. Eng Appl Artif Intell 20(2):147–161
16. Francisco GS, Rodrigo MB, Leonardo C, Jesualdo TFB, Dagoberto CN (2006) An ontology-based intelligent system for recruitment. Expert Syst Appl 31(2):248–263
17. Burstein MH (2004) Dynamic invocation of semantic web services that use unfamiliar ontologies. IEEE Intell Syst 19(4):67–73
18. Lee CS, Kao YF, Kuo YH, Wang MH (2007) Automated ontology construction for unstructured text documents. Data Knowl Eng 60(3):547–566
19. Chrissis MB, Konrad M, Shrum S (2005) CMMI: guidelines for process integration and product improvement. New York, Addison–Wesley
20. Staples M, Niazi M, Jeffery R, Abrahams A, Byatt P, Murphy R (2007) An exploratory study of why organizations do not adopt CMMI. J Syst Softw 80(6):883–895
21. Huang SJ, Han WM (2006) Selection priority of process areas based on CMMI continuous representation. Inf Manag 43(3):297–307
22. Sommerville I (2004) Software engineering. New York, Addison–Wesley
23. Liu X, Kane G, Bambroo M (2006) An intelligent early warning system for software quality improvement and project management. J Syst Softw 79(11):1552–1564
24. Greif N (2006) Software testing and preventive quality assurance for metrology. Comput Stand Interfaces 28(3):286–296
25. Voas J (2003) Assuring software quality assurance. IEEE Softw 20(3):48–49
26. Wooldridge M (2002) An introduction to multiagent systems. Chichester, Wiley