Journal of Software Engineering

ISSN 1819-4311

Academic Journals Inc.

www.academicjournals.com

Journal of Software Engineering 4 (1): 30-43, 2010
ISSN 1819-4311

Quality Measurement Model for KADS-Domain Knowledge

[1]A. El-korany, [2]D. Nabil and [3]A.S. Eldin
[1]Department of Computer Science, Faculty of Computers and Information, Cairo University, Egypt
[2]Department of Information Systems, Modern Academy, Maadi, Egypt
[3]Department Information Systems, Faculty of Computers and Information, Helwan University, Egypt

Abstract: This study proposes a quality measurement model for KADS-based knowledge base. Different quality models for object oriented design were investigated and are partially used as an initial step to develop quality models for KADS-domain knowledge. To assess the effectiveness of the proposed quality model a supportive automatic tool was developed and applied on a sample of real world agriculture domain knowledge developed by the Central Laboratory of Agriculture Expert System (CLAES).

Key words: Quality measurement model, quality metrics, knowledge-based systems, KADS, model validation

INTRODUCTION

The demand for analyzing expert systems to evaluate and measure their quality is becoming increasingly important as the paradigm continues to increase in popularity. Recent competition would have strongly concentrated on producing quality expert systems. The quality of knowledge-based system involves how well relevant parts of the application domain are captured by the knowledge base. AI communities have been agreed that the quality of expert systems is adequate to the quality of its stored knowledge (Preece and Shinghal, 1994). It has been found that existing Knowledge-Based Systems (KBSs) quality methodologies don't provide a formal judgment about the extent of success or failure of those KBSs. In particular, they lack referenced measurement methods and techniques for specified KB quality properties and metrics that would allow their wide acceptance by AI quality communities as indicated by Bruntink and Deursen (2004), Hendriks (1999), Vermesan (1997) and Clarke and Soltan (1995, 1996). Recently, Ontology has been used to model the content of domain knowledge. Researchers focus on applying software metrics to measure the quality of ontology as discussed by Orme *et al*. (2006, 2007).

Nowadays, the switch to representing complex KBSs through using Object Oriented (OO) approach offers the ability to apply the set of referenced quantitative and well-defined quality properties and metrics that are used to assess object oriented programs. From this perspective, the objective of this research is to propose a quality model for measuring the quality of KADS- based expert systems. KADS (Knowledge Acquisition and Development Systems) development methodology that has been developed by Edwards (1991) and updated to Commen KADS by Schreiber *et al.* (2000) lends itself to object oriented paradigm. KADS has been used to model ES in terms of three types of knowledge, namely, domain knowledge, inference knowledge and task knowledge. This study aims to providing a quality measurement model for KADs domain knowledge.

Corresponding Author: Abeer El-korany, Dr. Ahamed Zoweil Street, Orman, Giza 12613, Egypt

Domain knowledge is the factual knowledge about the application domain. It is structured internally into two parts: domain ontology and domain relations. Two key factors are the base for constructing such a model. First, established quality attributes properties and metrics in OO and software engineering had been revised and extended to enable the measurement process of common features that exist in both OO and KADS-domain knowledge. Second, a set of new quality properties and metrics to measure the characteristics that are particular to KADS-domain knowledge is proposed. The model is assessed by developing a tool that was used to statically measure the quality of a set of real world domain knowledge bases. The purpose of applying the experiment is to provide some evidence about suggested quality metrics and to ensure that obtained quality attribute values are within a accepted range.

THEORETICAL BACKGROUND

This study, provides a survey of well-known software and object oriented quality models. Then, it presents the current state of art of quality measurement research work that has been applied to KBSs.

Software Quality Models

McCall *et al.* (1977)'s quality model was one of the first well known quality models that aimed towards the system developers and the system development process. It his quality model, McCall *et al.* (1977) attempts to bridge the gap between users and developers by focusing on a number of software quality factor that reflect both the users' views and the developers priorities. The McCall quality model has three major perspectives for defining and identifying the quality of a software product: product revision, product and product operations. The model furthermore details the three perspectives in a hierarchy of factors, criteria and metrics. The quality factors describe different types of system behavioral characteristics and the quality criterions are attributes to one or more of the quality factors. The quality metric, in turn, aims to capture some of the aspects of a quality criterion

ISO/IEC 9126 (2001) standard defined software quality, which is described as using internal and external software qualities and their connection to attributes of software in a so-called Software Quality Model (SQM). The Software Quality Model defined in ISO 9126 follows the Factor-Criteria-Metric model proposed by McCall (1977). It defines six quality factors, which are refined into criteria. These criteria are in turn assessed by metrics measuring the design and development process and the software itself. The ISO 9126 quality factors are Functionality, Reliability, Usability, Efficiency, Maintainability and Portability. These factors are further subdivided into sub characteristics such as suitability, accuracy, security, compliance, time behavior, adaptability and so on. These sub characteristics can be measured by internal or external metrics. The standard claims that these six characteristics are comprehensive; that is, any component of software quality can be described in terms of some aspect of one or more of the six factors. Some attributes are in conflict with each other. Therefore, the customer and the software developer must work together to define which attributes are essential to a particular project.

Object Oriented Quality Models

Recently, object-oriented paradigm has provided other elements that are used to assess software quality (Shalloway and Trott, 2002) From this perspective, quality must be changed to rely on some specific properties such as encapsulation, inheritance and polymorphism etc. This has led to the definition of new metrics for OOP measurements like follows:

Chidamber and Kemerer (1994) introduced the Metrics for Object Oriented Software Engineering (MOOSE metrics). This metrics suite consists of six metrics that assess different characteristics of OO: Weighted Methods per Class (WMC) that can be classified as a traditional complexity metric, Depth of Inheritance Tree (DIT) of a class that can be classified as inheritance metric, Number Of Children (NOC) that can be classified as hierarchy metric, Coupling Between Object (CBO) classes that can be classified as coupling metric, Response For a Class (RFC) that can be classified as coupling metric, Lack of Cohesion in Methods (LCOM) that can be classified as cohesion metric.

Metrics for Object Oriented Design (MOOD) launched by Abreu and Carapuça (1994) was one of the earliest quality models for OOP. Metrics for object oriented design objective was to enable identifying quality measurement criteria for object-oriented designs by means of quantitative evaluation (i.e., metrics) of object-oriented paradigm. This was expressed in the form of internal as well as external characteristics. The original MOOD set considered of eight metrics, each of these metrics refers to a basic structural mechanism of object-oriented paradigms. Metrics for object oriented design embedded in a quality model by Abreu and Melo (1996), empirically validated by Basili *el al.* (1996). A newer version (MOOD2) and MOODKIT, a metrics collection automation tool, were modified also by Abreu and Cuche (1998). A formal method for representing MOOD2 metrics using object constraint language discussed by Baroni *et al*. (2002).

Recently, Bansiya and Davis (2002) suggested a Quality Model for Object Oriented Design (QMOOD) that defined the lower level design metrics in terms of design characteristics and quality is assessed as an aggregation of the model's individual high-level quality attributes. The suggested design quality attributes are (reusability, flexibility, understandability, functionality, extendibility and effectiveness). While, the suggested design quality properties are (design size, hierarchies, abstraction, encapsulation, coupling, cohesion, composition, inheritance, polymorphism, messaging and complexity). Finally, a set of quality metrics' for linking adjacent levels (relating a lower level to the next higher level) are provided.

Other OO metrics have recently been proposed in the literature like that proposed by Briand and Wust (2001) and Nejmeddine (2002, 2005). However, most of them are built upon the original Chidamber and Kemerer (1994) metrics suite and devoted for empirically validating and linking them to OO quality attributes.

Knowledge-Based System Quality

The demand for quality measuring methods for KBSs provided the impetus for several well-funded research effort to tackle aspects of this problem. These researches concluded that the distinct characteristics of KBS require quality measuring methods different from those of conventional software. Interestingly, certain aspects of software quality measurement seem to be easier to measure for KBS; other aspects, however, seem to be much harder for KBS (Clarke and Soltan, 1995, 1996). This section presents some of the work in that area.

Groot *et al*. (2005) proposed a quantitative analysis for the robustness of KB and its components based on the idea of degradation tests: analyze how the quality of the output degrades as a function of degrading input. He suggested two quality metrics for output quality that are: recall (is the fraction of correct answers that the system actually computes) and precision (is the fraction of computed answers that are actually correct). Recall correspond to the degree of the completeness of the system where precision corresponds to the degree of the soundness of the system. Completeness of the input is the number of available observations. He also defined other set of notions that can be used for comparing

robustness of systems which are: monotonically, quality value, rate of quality change and integral of quality value. Based on these robustness definitions, the system can compute a score for every output class based on the given input. Scores are adjusted incrementally when new observations are added, the user see the interpretation of these scores.

Kramer and Kaindl (2004) presented a quantitative model for deviating two metrics suite for KBS. These matrices are coupling and cohesion. Their quantitative model is partly based on measurement theory of Fenton and Pfleeger (1997) that begin with identifying the attributes of interest, then deriving metrics for these attributes. The derivation process starts with the empirical relational system that consists of an intuitive ordering of the entities of the real world to be measured (in their context, frames and rules). Then, the relations established in the empirical relational are formalized by mapping them to the formal relational system. Coupling metric has been defined as the degree of interdependence between frames. While, cohesion metric represents the dependency degree between slots of a frame. Cohesion matrix was defined as the percentage of edges that actually occur from the edges that could possibly exist.

Ontology quality analysis Nowadays, ontology is widely used to model the content of domain knowledge. Some researchers focus on applying software metrics to measure the quality of ontology. Burton-Jones *et al*. (2005) provided a suite of metrics for analyzing ontology quality. These matrices aimed to measure syntactic quality, semantic quality, pragmatic quality and social quality, using sub-attributes such as lawfulness (correctness of syntax in terms of markup or harm style errors), richness (number of different properties used within an ontology), interpretability (meaningfulness of terms in relation to WordNet), consistency (whether a term is used in more than one way within an ontology), clarity (number of word senses of each word in an ontology), comprehensiveness (number of classes and properties), accuracy (checking knowledge in an ontology versus knowledge known to be), relevance (number of statements in the ontology that are used), authority (average number of links between ontologies within a library) and history (number of times an ontology has been used). In all cases they primarily consider the linguistic aspects of ontology. Recently, Orme *et al.* (2006, 2007) examined coupling cohesion and complexity of ontology deployed as a part of a software system. They examine how changes in ontology metrics as the ontology evolves could affect ontology stability, completeness and domain focus (evolution of ontology).

The main weakness of most of the earlier study lies in fact that they do not really bring confidence in their metrics. Currently, many of these quality attributes are answered by indirect evidence only without any obligation proof as discussed by Cairo *et al*. (2000).

MODEL DEVELOPMENT

The proposed Quality Measuring Model (QMMK) for KADS-domain knowledge has been developed based on ISO/IEC9126-1standard (2001) for software product quality measurement model as well as Bansiya *et al*. (2002) Quality Model for Object Oriented Design (QMOOD).

Structure of Quality Model

Dormey (1995)'s generic quality framework is a methodology for the development of quality model in a bottom-up fashion. It relied on the decomposition of high level quality attributes into tangible, quality-carrying properties of software product components. There are three main principal elements to Dromey's generic quality model: product property that influence quality, a set of high level quality attributes and a mean of linking them. The

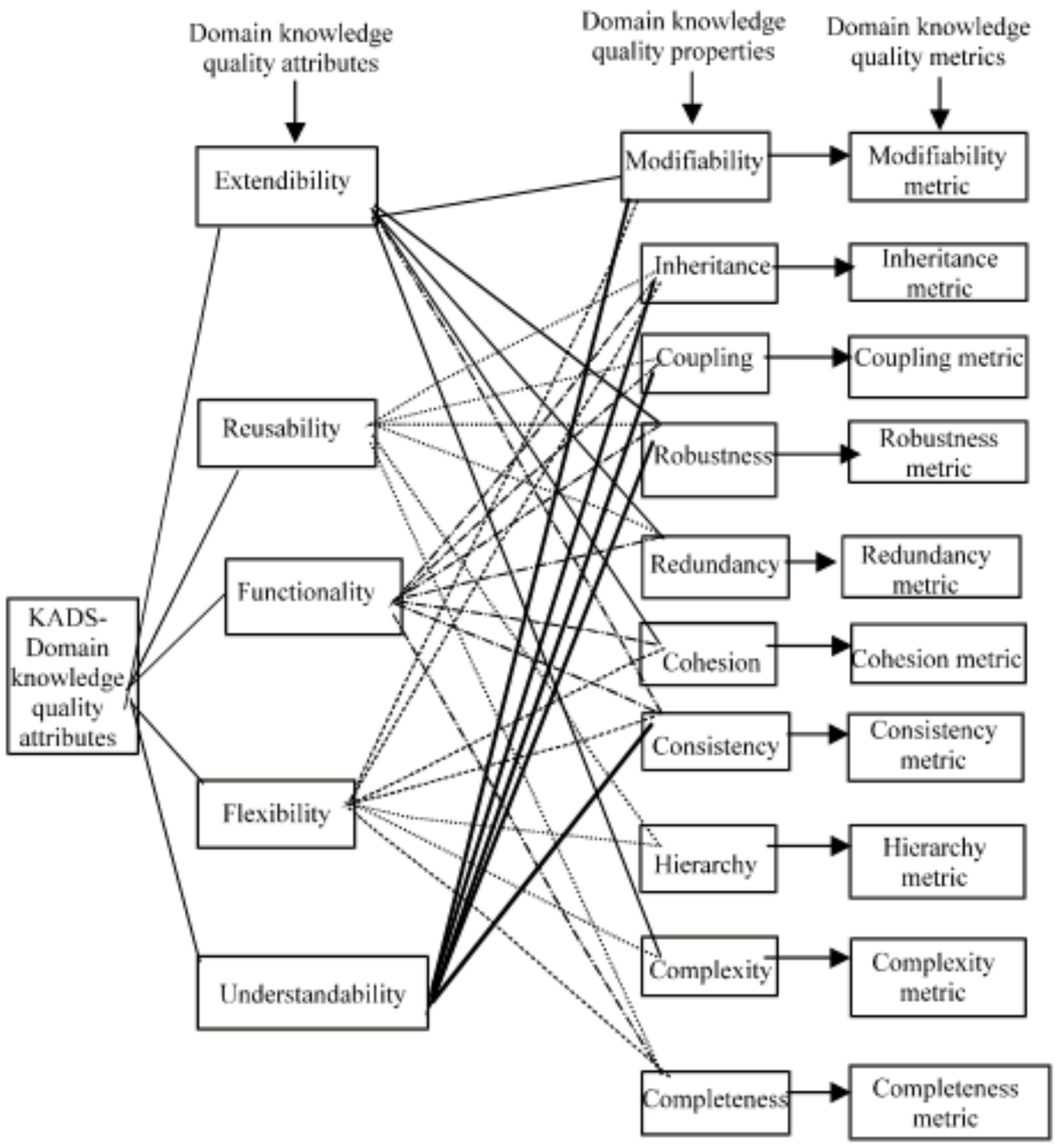

Fig. 1: KADS-Domain knowledge quality model

proposed model applies the same bottom up mechanism and focuses on defining domain knowledge quality attributes and domain knowledge quality properties based on QMOOD and ISO 9126 quality frameworks. Then it attempts to link these quality attributes and properties together. Figure 1 show the three levels that constitute this model:

- Defining KADS-Domain knowledge quality attributes
- Defining KADS-Domain Knowledge quality properties
- Linkage (metrics) that connect quality properties with quality attributes

Each of these levels is described in the following sub-sections

Defining KADS-Domain Knowledge Quality Attributes

The ISO 9126 quality attributes functionality, reliability, efficiency, usability, maintainability and portability were selected as the initial set of quality attributes of the proposed KADS-domain knowledge quality measuring model. These attributes were individually reviewed to find out if they contributed towards the nature of domain knowledge and whether this set is sufficient broad to include all quality aspects of domain knowledge. Some of these attributes such as reliability, efficiency, usability and maintainability were excluded due to their obvious slant toward implementation and sustainability of the product under specific conditions. This is beyond the scope of statically measuring quality of KADS-domain knowledge. The term extendibility is better than portability in presenting the degree of new incorporation of new inputs in domain knowledge structure.

Flexibility of domain knowledge is also an important characteristic that supports incorporation of changes in domain knowledge structure to provide function related capabilities. Therefore, it was decided to include flexibility as a quality attribute in the proposed model. The capability of domain knowledge to be simply reapplied to a new application without a significant effort and easily learned by the developers justified the inclusion of reusability and understandability, respectively as an important quality attributes of KADS-domain knowledge. Thus, the initial set of quality attributes of KADS-domain knowledge shown in Table 1 are: reusability, flexibility, extendibility, under standability and functionality as. This set of quality attributes is broad enough to allow desirable quality attributes of KADS-domain knowledge to be identified. These proposed quality attributes are not exclusive and it can be easily changed to represent different objectives and goals.

Defining Domain Knowledge Quality Properties

The second step of proposed KADS-Domain Knowledge quality measuring model is to provide a set of KADS-Domain Knowledge quality properties. KADS development methodology used to build the knowledge base lends itself to object oriented paradigm. Therefore, identification of KADS-Domain Knowledge quality properties extends OO and ISO 9126 software quality properties. Due to the distinct features of KADS-domain knowledge, usage of some OO quality properties is limited and a new set of KADS-Domain Knowledge quality properties is previously suggested by Nabil *et al*. (2005) and described in Table 2. Similarity between KADS domain knowledge and OO allow the usage of some properties like inheritance, hierarchy and coupling. These properties represent the

Table 1: Domain knowledge quality attributes definitions

Quality attribute	Description
Reusability	Reflects the presence of domain knowledge characteristics that allow it to be reapplied to a new problem without a significant effort.
Flexibility	Characteristics that allow the incorporation of changes and modifications in the domain knowledge structure. The ability of domain knowledge to be adapted to provide functionally related capabilities.
Extendibility	Refers to the capability of the domain knowledge structure to allow for the new incorporation of new components in the structure
Understandability	Reflects the capability of domain knowledge that enables it to be easily comprehended through understanding of its components and structure.
Functionality	The capability of each domain knowledge components to provide functions that meet the needs of other knowledge components

Table 2: Proposed set of quality property for KADS-domain knowledge

Quality properties	Description
Hierarchy	Reflects the presence and usage of different generalization- specialization component in the domain knowledge structure. It is the count of the number of non-inherited classes that have children.
Inheritance	It is reflected the relationship level between components in the domain knowledge hierarchy tree. This relationship is related to the level of nesting of domain knowledge concepts in an inheritance hierarchy.
Coupling	Reflects the interdependency of one component on another component of the domain knowledge structure.
Complexity	A measure of the degree of difficulty in understanding the internal and external structure of domain knowledge component.
Redundancy	A redundant is found when it is possible to reach the same conclusion from the same inputs.
Cohesion	Assess the relatedness of component contents. Strong overlap in component input and output is an indication of strong cohesion.
Modifiability	The capability of domain knowledge component structure to be modified.
Robustness	The capability of domain knowledge component to function correctly in the presence of internal contradiction of component's structure
Completeness	The ability of domain knowledge to be complete in content and structure.
Consistency	The capability of domain knowledge component to provide no internal conflict.

dependency of one KB component on another one. Other related properties like cohesion that expresses the amount of relatedness among system components, while complexity corresponds to the number of each component inputs and outputs. Particular features of KADS-domain knowledge require definition of some quality properties such as consistency between KB structure, robustness of domain knowledge components against invalid inputs, response to knowledge modifications, as well as redundancy and completeness.

Identifying Domain Knowledge Quality Metrics

A survey of existing ISO 9126, OO quality metrics revealed that there are several metrics have been defined and could be directly applied in measuring KADS-domain knowledge quality such as hierarchy, inheritance and coupling quality metrics. Also, there are several other existing quality metrics such as cohesion, consistency and complexity that need to be modified to suite the characteristics of domain knowledge. On the other hand, there are other quality properties, such as modifiability, completeness, complexity that need to be modified to suite the characteristics of domain knowledge. On the other hand, there are other quality properties, such as modifiability, completeness, redundancy and robustness, for which no quality metric exist. These metrics require a complete definition before they can be used in the proposed KADS-Domain Knowledge quality model. The complete set of matrices that is suited to different domain knowledge components.

MAPPING QUALITY ATTRIBUTE/PROPERTY FOR KADS DOMAIN KNOWLEDGE COMPONENTS

As described earlier in this study, the KADS development methodology is used to model KBSs in terms of three types of knowledge, namely: domain knowledge, inference knowledge and task knowledge. Task knowledge describes a fixed strategy for solving a problem. Inference knowledge describes declaratively, the operations that can be made using domain knowledge and the role that this knowledge can play in the reasoning process. Domain knowledge is the factual knowledge about the application domain. It is structured internally into two parts: domain ontology and domain relations. Domain ontology is the most fundamental part of KBS and it includes domain concepts declarations that represent the basic terminology used by applications in a specific domain. Each concept is identified through its name and described in details in terms of properties. Each property is also described by facets, which specify its data type, legal values and source of value (Rafea *et al*., 2003). Domain relations represent different interactions between these domain concepts and their properties. These relations are built on the top of the declared concepts, in form of rules, tables and mathematical functions.

The following subsection, describes how the proposed quality model is used to measure the particular characteristics of each KADS domain knowledge components.

Mapping Property/Metrics to Domain Ontology

The domain concept is the most fundamental part of domain ontology that represents the shared vocabulary used through all other domain knowledge components. Elements that constitute the domain concepts like properties, facets and values of these facets have similar nature to those of class component in OO. Therefore, some class quality metrics like depth of inheritance (DIT), Coupling Between Objects (CBO), Number of Roots(NOR) that are used in OO to measure inheritance, couplingand hierarchy quality properties, respectively, are used in the assessment of domain concept quality. On the other hand, a lack of communication through message passing which is not supported by domain concepts,

affects applying of some Chidamber's OO quality metrics such as Response For Class (RFC), Weighted Methods per Class (WMC) and cohesion (LCOM).

Furthermore, other class quality properties cannot be directly applied to domain concepts and need to be adapted to suite its specific nature. For example complexity quality property was used to measure the degree of understanding the internal and external structure of class is replaced by consistency and coupling. Since consistency represents the internal homogeny within parts of concept, while coupling cover concept interdependency in concepts hierarchy structure. Consistency quality property of domain concept is characterized through exposing the internal contradiction between the same concepts-property pair and their facets (For example, if concept type facet is numeric this means concept legal value facet must have an upper and lower range, where the nominal facet will have a text legal value). While, coupling measures the percentage of inherited properties in a concept from its supper concept.

An important point to note here that the nature of knowledge allow the incorporation of recent requirements and changes, which indicate the high significant of introducing specific quality property for measuring the ability of domain ontology to respond to any change. This led to the definition of two new DKQMM quality properties and metrics: Concept Modifiability (CM) and Concept Completeness (CC) which are represented in Table 3.

Mapping Property/Metrics Quality to Domain Relations

Domain relations represent different interactions between domain concepts and their properties in the form of rules, tables and mathematical functions. The meanings implied by the domain relations are different from classes implied by OO. Thus, the measuring properties and metrics for OO cannot be directly applied to domain relations due to their specific nature. Rather, some of OO quality properties are adjusted to be consistence with the structure of KADS-domain relations. This adjusted set of properties which are described in Table 4 includes: coupling which represent the dependency of one relation on another, complexity: the number of relations' inputs and outputs pair and cohesion which correspond to the amount of relatedness among relations which has been discussed by El-Emam *el al.* (2001). Cohesion metric measure the percentage of input pairs of each individual relation w.r.t the total number of input pairs that is used in the same relation cluster. This is achieved using the medium of jebetri to get the cohesion degree. Other quality properties of domain relations are introduced in order to measure particular features of domain relation such as consistency, robustness, modifiability effect and redundancy.

Table 3: The quality properties/metrics for domain ontology

Property	Property metric	Name	Description
Inheritance	DIT	Depth of Inheritance Tree	This metric is a count of the total number of node concepts that their edges exceed three levels relatively to the root of the hierarchy tree.
Hierarchy	NOR	Number of Root concepts	This metric is a count of the total number of root concepts that non-inherited and have children. These children can be classified to parents and ancestors of other concepts.
Coupling	NIP	Number of Inherited properties	Applying the inheritance vision of coupling, NIP is a measure of the percentage of inherited properties of a concept from its supper.
Consistency	COS	Consistency of Structure	The extent to which concept facets are valid with each other
Modifiability	CM	Concept Modifiability	It is a measure of the number of all domain knowledge components that can be potentially affected in response to any concept modification.
Completeness	CC	Concept Completeness	It is a measure of the ability of concept/properties/facets to be fulfilled in description, source of value, type and legal values.

Table 4: The quality properties/metrics for domain relations

Property	Property metric	Name	Description
Coupling	CBR	Coupling Between Relations	It is a measure of the number of relations that has an input part using an output pair of other relations (coupling between relations via I/O pairs).
Cohesion	RC	Relation Cohesion	It is the degree of relatedness between relations. It is measured through counting the number of repeated input and outputs pairs at all of relation's clusters.
Complexity	NOI	Number of Inputs	NOI is a count of relations that their inputs conditions exceed three input pairs.
Redundancy	RR	Relation Redundancy	It represents the count of relations that have the same inputs and the same outputs.
Modifiability	RM	Relation Modification	It is a measure of the number of the domain knowledge components (rules, functions and tables) that can be potentially affected in response to any domain relation modification.
Consistency	COR	Consistency of Relation	It is a count of undefined input/output parts of domain relation with respect to concept ontology.
Robustness	ROR	Robustness of Relation	Relation robustness is a number of relations that has invalid rules input pair (contradicted inputs).

Table 5: Domain relation quality attribute and their properties/matrices

Attribute/properties	Reusability	Flexibility	Extendibility	Understandability	Functionality
Coupling	✓		✓	✓	
Cohesion	✓			✓	✓
Complexity		✓	✓	✓	
Redundancy	✓		✓		✓
Modifiability	✓		✓		✓
Consistency		✓	✓	✓	✓
Robustness			✓	✓	✓

Linking Property/Metrics to Domain Knowledge Quality Attributes

This section presents a basis for relating quality properties of domain knowledge components to its quality attributes which are indicated in Table 5. The researchers earlierly suggested a set of computation formulas to measure the influence of quality properties/metrics on quality attributes of different domain knowledge components as appeared in Nabil *et al*. (2005, 2008).

Inheritance Metric

This metrics assesses the potential of reuse of a domain concept and its probable ease of understandability. Domain knowledge with small DIT has much potential for reuse and understands. On the other hand, as a concept gets deeper into a concept hierarchy, it becomes more difficult to be flexible, due to the increase of mental burden needed to capture its functionality.

Hierarchy Metric

When the number of children of a concept grows, this indicates the less significant of that concept to be reused by another application. Furthermore, the huge number of children leads to improper flexibility of the parent and complicate it. Low number of children is considered good for high functionality and extendibility.

Coupling Metric

The more independent a domain concept is, it is easier to be reused in another application (i.e., when coupling between concepts decrease, the reusability of that concept increase). Also, the larger the number of inherited properties between concepts, the higher the sensitivity to apply changes in the system and therefore extendibility is slightly difficult.

Strong coupling complicates a system and make it harder to understand. Therefore, for domain knowledge, coupling affects reusability, extendibility and understandability.

Robustness Metric

In general, robustness quality property is the degree to which a concept can function correctly in the presence of an internal contradiction between elements. An important point to note here is that high robustness is viewed to promote reusability in another application. While a low robustness provides good results for understandability, functionality and extendibility.

Completeness Metric

This metric assesses the universal applicability of domain knowledge since it measures gaps of domain knowledge. For domain concept (for example), it reflects how each concept/properties/facets are fulfilled in description, source of value, type and legal each values. Completeness is viewed to promote reusability and functionality. We can note that high measures of completeness are viewed to influence positively these quality attributes.

Modifiability Metric

Domain knowledge components should be organized in such a way as to permit modification operations to be carried out. Modifying a domain knowledge component implies that we are able to add, edit and delete many instance of this component. Applying either of these operations may affect other domain knowledge components that use this modified component. So, it is clear that high modification capability of domain knowledge viewed to promote understandability, extendibility and flexibility.

Cohesion Metric

Cohesion refers to the internal consistency within domain knowledge and how its content is related to provide a bounded functionality. For domain relation, cohesion can be measured as the degree of relatedness between relations. Low cohesion increases complexity and thereby decreasing understandability. High cohesion promotes reusability and functionality.

Redundancy Metric

Redundancy reflects the capability of domain relations to reach the same conclusion from the same inputs. Existence of redundant domain knowledge component increases the size of domain knowledge and therefore makes it more complex to understand. Redundancy decreases the capability of domain knowledge to be reused by other applications. Furthermore, domain knowledge with low redundancy provides more potential for extendibility and increase the capability to capture its functionality.

Complexity Metric

The key issue in complexity metric is the measure of the difficulty in understanding and managing the domain knowledge structure. As the size of each domain knowledge component increase, it provides low indication of understandability and become harder for reuse. Low complexity degree is considered good for high extendibility and flexibility.

Consistency Metric

Consistency metric reflects the freedom of domain knowledge components from internal contradiction. High consistency is viewed to promote reusability with other applications and influence positively on understandability, extendibility, functionality and flexibility.

MODEL VALIDATION FRAMEWORK

To validate the set of metrics used in the proposed quality QMMK model, Kitchenham *et al*. (1995) validation framework for evaluating software metrics is applied. In this framework, they described the structure of any measure as containing the entities being analyzed such as concept and relation; the attribute being measured, such as hierarchal structure; the unit used, such as number of concepts; and the data scale: nominal, ordinal, interval, or ratio. Units are valid only for interval. In order for a value to have any meaning, the entity, the attribute being measured and the units must be specified. The measure must be defined over a specified set of permissible values (discrete or continuous). In order to be valid, a measure must have:

- Attribute validity-the entity being analyzed has the attribute
- Unit validity-the unit is appropriate for the attribute
- Instrumental validity-the underlying model is valid and the instrument was calibrated
- Protocol validity-the protocol used for the measurement was valid and prevented errors such as double counting

Applying these measurements to the set of proposed metrics, we conclude the following:

Table 6: Validation of the proposed metrics

Metric	Entity	Quality property	Measure unit	Data scale (ratio)	Non-negativity /normalization	Null value	Monotonicity
DIT	Domain concepts	Inheritance	Length between each node concept to its root concepts	*	*	*	*
NOR	Domain concepts	Hierarchy	Number of root concepts	*	*	*	*
NIP	Domain concepts	Coupling between concepts	Inherited node Concept-property -facet-value	*	*	*	*
CBR	Domain relations	Coupling between relations	depended Input/ output pair of each relation	*	*	*	*
COS	Domain concepts	consistency	Root Concept-property-facet-value	*	*	*	*
CM	Domain concepts	modifiability or relations	No. of modified Concept-property-facet-value	*	*	*	*
RM	Domain relations	modifiability	No. of modified pairs of each relation	*	*	*	*
CC	Domain concepts	Completeness	No. of fulfilled concept/property/facet	*	*	*	*
RC	Domain relations	Cohesion between relations	Repeated Input/output pair of each relation	*	*	*	*
NOI	Domain relations	complexity	No. of input pair/relation	*	*	*	*
RR	Domain relations	redundancy	Redundant Input/ output pair between relations	*	*	*	*
COR	Domain relations	Consistency of relations	No. of undefined input/output pair of each relation	*	*	*	*
ROR	Domain relations	Robustness between relations	No. of contradicted inputs pair	*	*	*	*

*Indicates the value that vary from 0-1 depending upon the system being measured

- The entity is domain knowledge
- The data scale is interval ration
- All matrices have instrumental validity since the automatic tool perform the appropriate count correctly (as indicated by the case study in the next section)
- All matrices have protocol validity meaning that the measurement as defined in the formal notation is consistent and prevent double counting
- All matrices have unit and attribute validity

The analysis shown in Table 6 that indicates QMMK metrics meets Kitchenham *et al*. (1995) evaluation framework and meets the non-negativity, normalization and null value evaluation properties as well. The value of these metrics is always at least zero and always falls within the range (0, 1).

CASE STUDY

A supportive automated tool was developed and applied on a sample of real world domain knowledge bases in order to provide some evidence about the effectiveness of the proposed domain knowledge quality model. This tool was built using C no and is able to comprise domain knowledge component identification, metrics definition and calculation, then presenting the result in a report format. According to the design of QMMK no tool, domain knowledge components is selected from a pull-down menu that contains all the names of different domain knowledge. Then, the user is free to apply the required measurement criteria. Finally, a report containing the value of the required quality metric and each quality attribute is generated. The tool was applied on a sample of four real world expert systems developed at The Central Laboratory for Agricultural Expert Systems (CLAES). These four examples demonstrate the capability of our proposed model to measure the quality of domain knowledge. The values are computed based on the application of a mathematical relationship between attribute and property/metrics of the domain relation as shown in Table 7 and 8, respectively. The result coincides with previous linking analysis between the properties and quality attributes of domain relation shown in Table 6. The result also validate that the computed attribute value is within a valid range. The researchers believe that approving the result by a set of domain experts will enrich this.

Table 7: Computed quality metrics for domain relations

	System			
Metric	1	2	3	4
Coupling	0.10	0.23432	0.04515444	0.11124441
Cohesion	0.66667	0.3213211	0.55	0.53222345
Complexity	0.7443223	0.44566665	0.61111425	0.83454543
Redundancy	0.00	0.00	0.00	0.00
Modifiability	0.51843	0.7631	0.69249	0.84217
Consistency	0.94444	1.00	1.00	0.973713
Robustness	0.9877	1.00	1.00	1.00

Table 8: Computed quality attribute for domain relations

	System			
Domain relations quality attributes	1	2	3	4
Reusability	0.716991925	0.777493613	0.746010048	0.682515357
Flexibility	0.80226875	0.892425138	0.909350695	0.96056388
Understandability	0.518974425	0.609293338	0.591577133	0.507742841
Functionality	0.9502975	0.682410825	0.90121139	0.858212986
Extendibility	0.625894713	0.796551669	0.764211414	0.78971777

CONCLUSION

In summary, this study presented a model for measuring the quality of KADS domain knowledge. This model is based on hieratical Bansiya quality model for object oriented design that used a set of quality metrics as indicators to different quality attributes. The proposed model applies the measurement criteria to measure the common features between domain knowledge and OO and proposes a set of new quality properties/metrics to measure the particular characteristics of domain knowledge components. The model is enriched by a theoretical validation framework that aims to provide some evidence about the suggested quality metrics. A tool was applied on a set of real world agriculture domain knowledge to ensure that attribute values are within an accepted range. The obtained results would provide better influence if it had been compared by a set of domain expert which is currently being applied. The model will be extended to support the quality of web-based domain knowledge bases. It is also intended to modify the model to support the quality measurement of well knowledge ontologies.

REFERENCES

Abreu, F.B. and R. Carapuça, 1994. Candidate metrics for object-oriented software within a taxonomy framework. J. Syst. Software, 26: 87-96.

Abreu, F.B. and W. Melo, 1996. Evaluating the impact of object-oriented design on software quality. Proceedings of the 3rd International Symposium on Software Metrics: From Measurement To Empirical Results, Mar. 25-26, Berlin, Germany, pp: 90-96.

Abreu, F.B. and J.S. Cuche, 1998. Collecting and analyzing the MOOD2 metrics. Proceedings of the Workshop on Object-Oriented Product Metrics for Software Quality Assessment (ECOOP'98), Jul. 21, Brussels, Belgium, pp: 258-260.

Bansiya, J. and C.G. Davis, 2002. A hierarchical model for object oriented design quality assessment. IEEE Trans. Software Eng., 28: 4-17.

Baroni, A.L., S. Braz, F.B.E. Abreu and N.L. Portugal, 2002. Using OCL to formalize object-oriented design metrics definitions. Proceedings of the 6th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, 2002, Malaga, Spain, pp: 1-11.

Basili, V.R., L.C. Briand and W.L. Melo, 1996. A validation of object oriented metrics as quality indictaors. IEEE Trans. Software Eng., 22: 751-761.

Briand, L.C. and J. Wust, 2001. Modeling development effort in object-oriented systems using design properties. IEEE Trans. Software Eng., 27: 963-986.

Bruntink, M. and A.V. Deursen, 2004. Predicting class testability using object-oriented metrics. Proceedings of the, 4th IEEE International Workshop on Source Code Analysis and Manipulation, Sept. 15-16, Washington, DC., pp: 136-145.

Burton-Jones, A., V.C. Storey, V. Sugumaran and P. Ahluwalia, 2005. A semiotic metrics suite for assessing the quality of ontologies. Data Knowl. Eng., 55: 84-102.

Cairo, O., J. Barreiro and F. Solsona, 2000. The importance of metrics and evaluation processes in knowledge based system. Proceedings of 2nd International Workshop on Computer Science and Information Technologies, Sept. 18-23, Ufa, Russia, pp: 246-255.

Chidamber, S.R. and C.K. Kemerer, 1994. A metrics suite for object oriented design. IEEE Trans. Software Eng., 20: 476-493.

Clarke, R. and H. Soltan, 1995. Towards quality control for knowledge based systems development. Knowl. Based Syst., 8: 269-277.

Clarke, R. and H. Soltan, 1996. A framework for the generation of project specific quality control for knowledge based system development. Expert Syst., 13: 41-54.

Dormey, G.R., 1995. A model for software product quality. IEEE Trans. Software Eng., 21: 146-162.

Edwards, J.S., 1991. Building Knowledge Based Systems, Towards a Methodology. 5th Edn., John Wiley and Sons, New York, ISBN: 9780470217566.

El-Emam, K., S. Benlarbi, N. Goel and S.N. Rai, 2001. The confounding effect of class size on the validity of object-oriented metrics. IEEE Trans. Software Eng., 27: 630-650.

Fenton, N.E. and S. Pfleeger, 1997. Software Metrices: A Rigorous and Practical Approach. 2nd Edn., PWS Publishing Company, Boston, ISBN: 0-534-95600-9.

Groot, P., A.T. Teije and F.V.A. Harmelen, 2005. Quantitative analysis of the robustness of knowledge-based systems through degradation studies. Knowl. Inform. Syst., 7: 224-245.

Hendriks, P.H., 1999. The organizational impact of knowledge-based systems: A knowledge perspective. Knowl. Based Syst., 12: 159-169.

ISO/IEC 9126-1, 2001. Software Engineering-Product Quality Part 1: Quality Model. International Organization for Standardization, Geneva.

Kitchenham, S., S. Pfleeger and N. Fenton, 1995. Towards a framework for software measurement validation. IEEE Trans. Software Eng., 21: 929-944.

Kramer, S. and H. Kaindl, 2004. Coupling and cohesion metrics for knowledge-based systems using frames and rules. ACM Trans. Software Eng. Methodol., 13: 332-358.

McCall, J.A., P.K. Richards and G.F. Walters, 1977. Factors in software quality. Vol. I, II, III, RADC Reports.

Nabil, D., A. EL-Korany and A.S. Eldin, 2005. Quality measuring model for KADS-based expert systems. Proceedings of the IASTED International Conference on Computational Intelligence, 2005, Canada, pp: 337-341.

Nabil, D., A. EL-Korany and A.S. Eldin, 2008. Towards a suite of quality metrics for KADS-domain knowledge. Expert Syst. Appli., 35: 654-660.

Nejmeddine, T., 2002. Object-oriented system decomposition quality. Proceedings of the 7th IEEE International Symposium on High Assurance Systems, 2002, Tokyo, Japan, pp: 230-232.

Nejmeddine, T., 2005. Predicting maintainability using object oriented system decomposition metrics. Proceedings of the International Conference on Software Engineering Research and Practice, 2005, Las Vegas, Nevada, USA., pp: 942-946.

Orme, A.M., H. Yao and L.H. Etzkorn, 2006. Coupling metrics for ontology-based systems. IEEE Software, 23: 102-108.

Orme, A.M., H. Yao and L.H. Etzkorn, 2007. Indicating ontology data quality, stability and completeness throughout ontology evolution: Research articles. J. Software Maintenance Evol: Res. Prac., 19: 49-75.

Preece, A.D. and R. Shinghal, 1994. Foundation and application of knowledge base verification. Int. J. Intell. Syst., 9: 683-701.

Rafea, A., H. Hassan and M. Hazman, 2003. Automatic knowledge acquisition tool for irrigation and fertilization expert systems. Expert Syst. Appli., 24: 49-57.

Schreiber, G., H. Akkermans, A. Anjewierden, R. De-Hoog and N. Shadbolt *et al*., 2000. Knowledge engineering and management: The common KADS methodology. MIT Press, USA., ISBN-13: 978-0262193009.

Shalloway, A. and J. Trott, 2002. Design Patterns Explained: A New Perspective on Object-Oriented Design. 2nd Edn., Addison-Wesley, New York, ISBN: 0321247140.

Vermesan, A.I., 1997. Quality assessment of knowledge based software: Some certification considerations. Proceedings of the 3rd International Software Engineering Standards Symposium, Jun. 1-6, Walnut Creek, CA., USA., pp: 144-154.