An ontology for quality management — enabling quality problem identification and tracing

H M Kim, M S Fox and M Grüninger

The TOVE Quality Ontology-VB is the formal representation (using First-order Logic) of terms, relationships, attributes, and axioms about quality which are generic beyond any specific quality domain. The assumption that quality is 'conformance to requirements' is used to decompose the quality domain into domains of measurement, identification, and traceability for further exploration; the TOVE Quality Ontology-VB is comprised of ontologies of these domains. An ontological engineering methodology comprised of motivating scenario, scope, and competency question statements, data model and axiom construction, and competency question/answer visualisation is demonstrated. The methodology is applied to develop the TOVE Traceability Ontology-VB. This ontology's representations are used to construct quality control applications enabling quality problem identification and tracing. They also enforce the properties of an entity that make it traceable — a novel and useful feature for quality control applications.

1. Introduction

Quality has become a corporate cliché in the business world, but as with most clichés, the term 'quality' is more anecdotally and less formally defined. Quality gurus like Juran and Deming have espoused the importance of quality from informal, experiential and philosophical perspectives. Quality tools, such as statistical quality control (SQC) and quality function deployment (QFD), are defined with more rigour-mathematics for the former, and rules for building the 'House of Quality' [1] for the latter — but only relate to a specific domain of quality. ISO 9000 and Baldridge Awards do address the quality domain from both formal and generic views, as these documents specify guidelines or requirements for an acceptable or excellent quality level for a generic company. The TOVE Quality Ontology, Version Beta (TOVE Quality Ontology-VB)[1] endeavours to be an even more formal representation of terms, relationships, attributes, and axioms about quality. Yet these representations are also generic beyond any specific quality domain. The goals of the TOVE Ontologies project of the Enterprise Integration Laboratory at the University of Toronto are to:

- provide a shared terminology,
- define precise and unambiguous semantics for the enterprise [2].

Thus the TOVE Quality Ontology-VB is generic to satisfy the first goal, and formal to satisfy the second.

According to Godfrey [3], the need for an organisation to better manage information about quality will become more emphasised with increasing advances in information technology. Since the TOVE Ontologies project builds constructs for all strata of knowledge representation [4] — i.e. implementation, logical, conceptual, generic, and application — the TOVE Quality Ontology-VB, used with the other TOVE ontologies, provides a comprehensive and integrated set of representations with which sophisticated, deductive decision-making can be made.

The TOVE Quality Ontology-VB is used to support decision-making because it provides representations needed to construct enterprise models for quality management. Enterprise models are information systems tools used for enterprise design, analysis, and operations. For design and analysis for example, the TOVE Quality Ontology-VB terms, relations, and attributes are instantiated to construct a populated model of a given enterprise's inspection process. Ontology axioms are then used to deduce additional facts about the enterprise. Terms, relations, and attributes represent the vocabulary for posing queries upon the model, and the axioms represent the semantics to answer these queries.

Given TOVE ontologies and a sufficiently populated enterprise model, it is possible to repeatedly pose queries and analyse them to perform enterprise analysis and design tasks such as:

- determining which products often have quality problems,

[1] In this paper, all references to VB denote Version Beta, e.g. TOVE Traceability Ontology-VB.

- tracing back to processes where these problems often originate,
- making recommendations to re-engineer those processes.

The ISO 9000 Quality Advisor [5] is a software tool for re-designing an enterprise to comply to ISO 9000. The tool applies a set of axioms representing the ISO 9000 requirements to a populated enterprise model and automatically evaluates whether the modelled enterprise complies with the requirements. The axioms are constructed using quality ontology representations.

The representations with which the TOVE Quality Ontology-VB is constructed are from the TOVE Core Ontologies — ontologies of activity, state, causality and time [6], resource [7], and organisational structure [8]. Representations are added into the TOVE Quality Ontology-VB only if they can be defined in terms of the data model and axioms of the TOVE Core Ontologies. The ontology results from decomposing the quality domain into more specific domains, each of which is formalised as an ontology of the TOVE Quality Ontology-VB. In section 3, the TOVE Traceability Ontology-VB is presented. In section 4, concluding remarks and further work that builds upon the TOVE Quality Ontology-VB are stated.

2. Decomposing the task of engineering the TOVE Quality Ontology-VB

TOVE Ontologies are designed by specifying competency questions which characterise tasks that an ontology-based system must support. Once an ontology is completed, it must be possible to pose competency questions as queries using ontology terms, relations, and attributes. Then competency questions can be answered — i.e. answers to queries are deduced — using ontology axioms. In engineering a generic quality ontology, an overall system competency question can be: 'What is the quality of a given product, process, or system of the enterprise?' However, this question is too broad. Thus it is decomposed into competency questions narrow enough in scope to motivate development of ontology representations. This decomposition also constitutes an abstract exploration of the quality domain.

2.1 What is Quality?

The official definition of quality recognised by international standardisation bodies is from the ISO 9000: 'Quality is the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs' [9]. This vague definition can be augmented with a manufacturing-based definition [10], as stated by Crosby [11]: 'Quality means conformance to requirements.' Combining ISO 9000 and Crosby's definitions of quality, the basic premise of the TOVE Quality Ontology-VB is the following:

A need can be decomposed into a set of requirements upon features and characteristics of a product or service, and if all these requirements are conformed to, then that need has been satisfied.

2.2 Decomposing a quality need

Figure 1 shows that as a quality need is decomposed into business questions, there are concomitant competency questions at each level of decomposition, which drive the engineering of the TOVE Quality Ontology-VB. It is possible to decompose a vague need such as the need for a safe car into numerous concrete, measurable requirements on features or characteristics such as the requirement on stopping distance. Boehm [12] explains that requirements are verified and validated:

- verification: 'Are we building the product right?'
- validation: 'Are we building the right product?'

A key assumption of the TOVE Quality Ontology-VB is that it its representations can be used for verification, but not validation — representations can be used to verify if a requirement is conformed to, but not whether it is an appropriate requirement. The ontology provides the terms, relations, and attributes to represent the hierarchy of a requirement's sub-requirements, and measurable requirements on a product's features and characteristics. It also provides the axioms to determine whether a given measurable requirement is conformed to. A user represents requirements, hierarchies, and requirement verification rules (e.g. 'For a given requirement to be verified, all sub-requirements of that requirement must be verified', or 'For a requirement to be verified, it is sufficient for 95% of all sub-requirements on one feature or characteristic of a given product to conform') using ontology representations.

2.3 Quality domains represented in the TOVE Quality Ontology-VB

Conformance of the example requirement on anti-lock brakes cannot be determined unless the stopping distance is measured. Di Franca [13] notes that: 'Things are investigated in physics insofar as it is possible to measure them, and not with the impossible goal of discovering their intimate essence.' As in physics, measurement is the root of quality management, as evidenced by Federal Express' quality philosophy of 'measure, measure, measure.' So the domain of measurement must be explored for the quality ontology.

Before measurement, an entity must first be identified as an entity to be measured. So, the domain of identification, particularly unique identification, must be explored for the quality ontology. Entities are identified and measured because variability inherently exists. Through measurement, variability that points to a quality problem is identified. The most primitive analysis technique required to correct a problem is traceability — the capability to trace, for

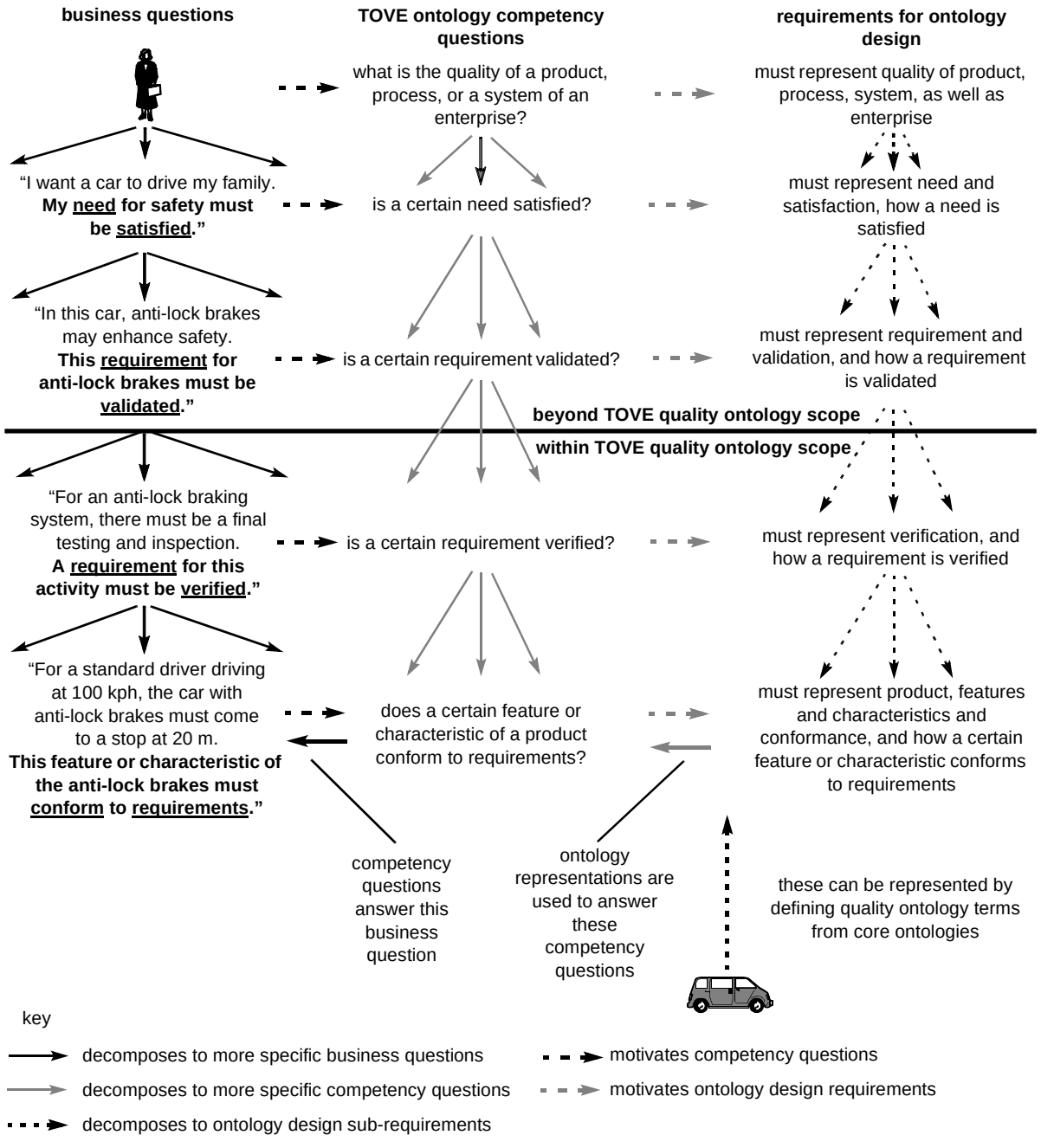


Fig 1 Engineering the TOVE Quality Ontology-VB — decomposing a need.

example, from a problematic assembly to its sub-assemblies to diagnose the root of a problem. So the domain of traceability is also explored.

The philosophy of minimal ontological commitment [14] bounds the scope of the ontology. There are more sophisticated quality analysis techniques than traceability, such as SQC, QFD, quality costing, and ISO 9000, which can be explored for the ontology. However, an ontological bias is introduced by committing to represent one or more of these analysis techniques to the TOVE Quality Ontology-VB. Therefore, only the most general quality analysis technique, traceability, is explored for the development of the TOVE Quality Ontology-VB, along with the general domains of measurement and identification. In particular, this paper details one of the ontologies which comprise the TOVE Quality Ontology-VB — the TOVE Traceability Ontology-VB. Since it is not discussed in detail in this paper, the reader is referred to the TOVE Measurement Ontology [5] for formal representations with which quality conformance can be evaluated. These representations comprise the TOVE Measurement Ontology-VB.

2.4 Structure of competency questions

For each domain explored for the development of the TOVE Quality Ontology-VB, the capability of the information system implementation constructed using the

ontology of that domain to answer competency questions validates the ontology. The competency of the ontology is determined by performing the following steps.

- Statement of motivating scenario

  This is a narrative about business issues and problems that ultimately an ontology-based system will address.

- Statement of scope

  Assumptions about the domain are made, and in so doing the scope of the ontology becomes more apparent. With these assumptions made, objects, relations, and attributes that belong in the ontology are vetted. Of the ontology evaluation criteria presented by Fox et al [2], the scope dictates the extensibility, granularity, and scalability of the ontology.

- Problem statement

  This is the one general problem statement that justifies the construction of the ontology. This is a question, rising from the motivating scenario and within the bounds of the scope of the ontology, posed generally to serve as the template for all competency questions. The problem statement affects the generality of the ontology.

- Statement of user competency questions

  Competency questions for the TOVE Quality Ontology-VB must be motivated by questions related to the motivating scenario. These questions are of the form of the problem statement, but more specific. These are called user competency questions, since these are types of question likely to be asked by the user of the ontology. These questions affect the competency of the ontology.

- Statement of developer requirements

  These questions characterise the design requirements of the ontology. These questions dictate the engineering of representations that are needed to answer user competency questions, and are likely to be asked by the developer of the ontology. These questions affect the efficiency, perspicuity, and transformability of the ontology.

3. TOVE Traceability Ontology-VB

In this section, sample representations from the TOVE Traceability Ontology-VB are presented to show the following:

- some representations required to represent the traceability domain,
- example of an application of the ontological engineering methodology,
- an example use of the ontology to solve a practical problem

The steps in the engineering of the Traceability Ontology-VB are included in the methodology shown in Fig 2.

3.1 Competency questions

Motivating scenario

The following motivating scenario demonstrates the importance of traceability for quality management.

- A quality problem with a batch of a final product is found. It is known that an electric surge occurred during a certain period of time, and that this may be the cause of the quality problem. For diagnostics, records of batches of different resources that were used or consumed during this time to eventually produce the

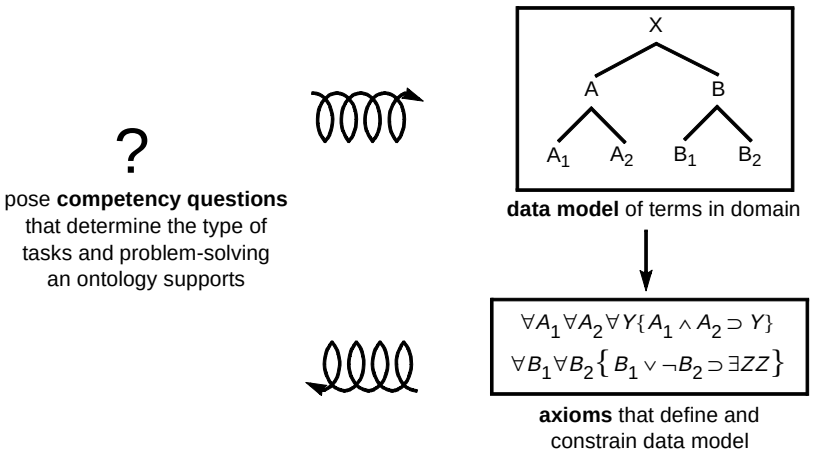


Fig 2 Ontological engineering methodology.

product must be examined. Also, records of batch quantities used or consumed may help, as would records of activities that used or consumed these batches.

Scope

Grady [15] states that traceability is a clear knowledge of ancestry, and so a discussion of traceability entails a graphical notation of ancestry — the tree. As such the task of traceability becomes one of recursively decomposing tree nodes, until terminal nodes are encountered. In the TOVE Core Ontologies, a non-terminal state is abstracted from a conjunction and/or disjunction of terminal states [2]. Therefore bounding scope entails determining acceptable traceability, given abstractions of conjunctive and disjunctive states. It must also be determined which of the TOVE entities are to be traceable.

TOVE Core Ontologies support abstraction of entities. For example, activities are comprised of sub-activities, each of which may have additional subactivities. It is possible to represent and reason only about the abstracted activities, or about all their hierarchy of sub-activities.

Given this, the following assumptions are stated to enable traceability of abstracted entities.

- Assumption 1

  It must be possible to trace from one entity to another, where neither of the entities are abstracted entities.

- Assumption 2

  If the previous assumption holds, it will be possible to trace from one entity to another, regardless of their levels of abstraction.

The following assumption determines which kinds of entities are traceable.

- Assumption 3

  Of the entities that the ISO 9000 recommends for tracing, only ‘product’ and ‘activity’ are to be uniquely identified and traced.

Combining these assumptions, what are appropriate terms for ‘product’ and ‘activity’ that are non-abstracted and traceable entities? ‘Traceable resource unit’ (tru) and ‘primitive activity’ are terms from the TOVE Identification Ontology-VB which are non-abstracted representations of ‘product’ and ‘activity,’ respectively. A tru is a ‘batch’ of a resource[2], and is not an abstraction of other resource-like entities; this term is more formally defined in the traceability ontology. A primitive activity is an activity[2] with no subactivities, and is also not an abstraction of other activity-like entities.

[2] This term is from the TOVE Core ontologies.

Problem statement

The problem statement is a general characterisation of the problem from the motivating scenario.

- Q: Given an entity and a state of the enterprise, can all entities and necessary attributes of these entities that had a bearing on the quality of the given entity be traced and identified?

The user competency question posed below is a form of this general problem statement.

User competency question

The following user competency question is directly motivated from the motivating scenario above.

- Q1: How much of a specific batch of a resource was used by one or more activities over a given period of time?

This question motivates the asking of more lower level questions that a developer is likely to ask, about how to model the traceability domain.

Developer requirements

In order to answer the user competency question, what characterises a traceable resource unit (a batch) must first be defined. For example, when can a traceable resource unit not be traced? So the following is asked:

- Q2: Under what condition(s) is traceability of a tru not possible?

What happens to the traceability of a tru if a portion of it is used or consumed for one activity, while the remaining portion is not? That is,

- Q3: What happens when a tru is split or disaggregated?

Conversely, what happens when two or more trus are brought together? That is,

- Q4: What happens when trus are aggregated?

What about quantity changes over time?

- Q5: Quantities of a tru will vary over time. How will this change be represented?
- Q6: Can the quantity of a tru be represented?
- Q7: Can the quantity of a tru at a certain point in time be represented?

3.2 Data model

Object-oriented model

The model (as depicted in Fig 3) represents the following:

- a tru is recognised to exist,
- a tru has a resource point or a quantity,
- there may exist a primitive trace between a tru and a primitive activity, a tru and another tru, or a primitive activity and another primitive activity,
- the trace between one tru/primitive activity pair to another tru/primitive activity pair is comprised of one or more primitive traces.

Terminological model

The object-oriented model can be re-expressed as a set of predicates with arguments.

- T1 — tru(Rt).
- T2 — primitive_activity(A).
- T3 — tru_known(Rt,Tp), where Tp is a time point at which point Rt is known to exist.
- T4 — rp_tru(Rt,Tp,Q,U), where Q is the quantity of Rt at Tp, measured in U units of measurement.
- T5 — primitive_trace(Rt_1,Rt_2,A,A,{L}); primitive_trace(Rt,Rt,A_1,A_2,{L}), where L is the primitive trace path between Rt_1 and Rt_2 via a common primitive activity A, or between A_1 and A_2 via a common tru Rt.
- T6 — trace(Rt_1,Rt_2,A_1,A_2,{L}), where L is the trace path between (Rt_1,A_1) to (Rt_2,A_2).

Next, axioms that formally define these predicates and constrain their proper use are presented.

3.3 Axioms

These are some of the axioms of the TOVE Traceability Ontology-VB, necessary to address developer requirements. For brevity, only some of the First-order Logic formalisations are shown.

- A1. tru(Rt): A tru is a homogeneous collection of one resource, that is used/consumed/produced/released[3] by a primitive activity in a finite, non-zero quantity.

$$\forall Rt \exists A \exists S \exists R \, \boldsymbol{tru}(\boldsymbol{Rt}) \equiv has\_tru(R,Rt) \wedge primitive\_activity(A) \wedge$$
$$((use(S,A) \wedge uses(S,Rt)) \vee (consume(S,A) \wedge consumes(S,Rt)) \vee$$
$$(produce(S,A) \wedge produces(S,Rt)) \vee (release(S,A) \wedge$$
$$releases(S,Rt))) \wedge$$
$$\forall Q(amount\_committed(S,Rt,Q) \supset Q>0).$$

where:

Rt is a traceable resource unit,

R is a resource, a collection of which comprises Rt,

S is state describing the use/consumption/production/release of Rt,

A: is the primitive activity which uses/consumes/produces/releases Rt,

and from TOVE Core Ontologies:

has_tru(R,Rt): resource R has a tru Rt,

use(S,A) ; uses(S,Rt): S is a state description for A's usage, but not consumption of Rt,

consume(S,A) ; consumes(S,Rt): S is a state description for A's consumption of Rt,

[3] These are terms from the TOVE core ontologies.

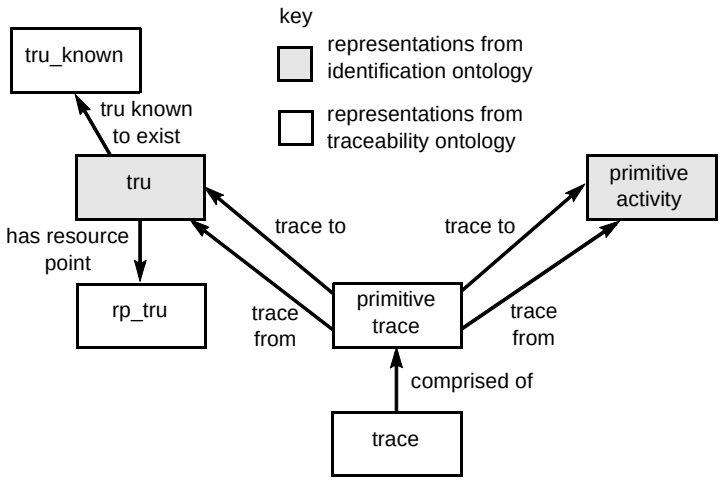


Fig 3 TOVE Traceability Ontology-VB data model.

release(S,A) ; releases(S,Rt): S is a state description for A's release of Rt,
produce(S,A) ; produces(S,Rt): S is a state description for A's production of Rt
amount_committed(S,Rt,Q): S is a state description of the amount Q committed to the consumption/usage/ /release/ production of Rt.

This axiom addresses all developer requirements.

- A2. tru_known(Rt,T_p): A tru is first 'known to exist' at the time that it is first used/consumed/produced/ released.

$\forall$ Rt$\exists T_p$ ***tru_known(Rt,T_p)*** $\equiv$ tru(Rt)$\wedge$
$\exists S_0 \exists T_0 \forall$ S$\forall$ T[(uses(S,Rt)$\vee$consumes(S,Rt) $\vee$produces(S,Rt)$\vee$releases(S,Rt))$\wedge$
(uses(S_0,Rt)$\vee$consumes(S_0,Rt)$\vee$produces(S_0,Rt) $\vee$releases(S_0,Rt))$\wedge$
(state_duration(S,T)$\wedge$state_durations(S_0,T_0) $\supset$T$\geq T_0$)$\wedge$
start_point(T_0,T_p)].

where:

Rt is a traceable resource unit,
T_p is the time point at which the tru is recognised to exist,
S is all states that use/consume/produce/release Rt,
S_0 is the first state that uses/consumes/produces/ releases Rt,
T,T_0 are time durations for states S, S0, respectively,

and from TOVE Core Ontologies:

state_duration(S,T): T is the time period covered by the state description S,
start_point(T,T_p): Tp is the start time point for T.

This axiom addresses developer requirements Q5, Q6 and Q7.

- A3. rp_tru(Rt,T_p,Q,U): If a tru is produced by a primitive activity, then the 'resource point of the tru' at time Tp (the time at which the tru is completely produced) is 'quantity that was produced',[4] Q, measured in U units.

$\forall$ Rt$\forall$ U$\forall$ Q$\forall T_p \exists$A $\exists$S
primitive_activity(A)$\wedge$ produce(S,A)$\wedge$
produces(S,Rt)$\wedge$tru_known(Rt,T_p)$\wedge$
amount_produced(Rt,Q)$\wedge\exists$R
unit_of_measurement(R,capacity,U,A) $\supset$
rp_tru(Rt,Q,T_p,U).

where:
Rt is a traceable resource unit,
Q is the quantity of Rt that was produced,
T_p is the time point at which Rt is produced,
U is the unit of measurement in capacity measurement units, e.g. 'obects' and tons,

[4] This phrase corresponds to the predicate amount_produced from the TOVE core ontologies.

A is the primitive activity that produces Rt,
S is the state associated with the produced tru Rt,
R is the resource which comprises Rt,

and from TOVE Core Ontologies:

amount_produced(Rt,Q): Q is the amount of Rt that was produced,
unit_of_measurement(R,capacity,U,A): Unit of measurement of capacity of a resource R which is used/consumed/ released/produced by A is U.

This axiom addresses developer requirements Q5, Q6 and Q7

- A4. rp_tru(Rt,T_p,Q,U): If a tru is released by a primitive activity, then the 'resource point of the tru' at time T_p (the time at which the tru is completely released) is the 'quantity committed to the activity',[5] plus the 'quantity that was already available to other activities',[6] measured in U units.

This axiom is used to address developer requirements Q5, Q6 and Q7.

- A5. rp_tru(Rt,T_p,Q,U): If a tru is used or consumed by a primitive activity, then the 'resource point of the tru' at time T_p (the time at which the usage or consumption is completed) is the 'quantity that was already available to other activities', Q, measured in U units.

This axiom addresses developer requirements Q5, Q6 and Q7.

- A6. rp_tru(Rt,T_p,Q,U): If there exists a 'resource point of a tru' at time T_{p1}, and there exists a 'resource point for the same tru' at time T_{p2}, and for any point T_p where $T_p \in (T_{p1},T_{p2})$ there does not yet exist a 'resource point for that tru,' then the 'resource point of the tru' for any point T_p is assigned to be the 'resource point of that tru' at time T_{p1}, because the quantity of the tru has not changed since T_{p1}.

This axiom addresses developer requirements Q5, Q6 and Q7.

- A7. Constraint on rp_tru: Before the time point, T_p, at which the 'tru is known to exist', there is no 'resource point of the tru.'

This axiom addresses developer requirement Q2.

[5] This phrase corresponds to the predicate amount_committed from the TOVE core ontologies.
[6] This phrase corresponds to the predicate amount_available from the TOVE core ontologies.

- A8. Constraint on rp_tru: However after T_p, there is always a 'resource point for the tru' (this value = 0, if the tru has been completely consumed).

  This axiom addresses developer requirement Q2.

- A9. Constraint on rp_tru: The 'resource point for a tru' is never incremented after it is recognised to exist.

  $\forall Rt \forall T_{pz} \forall Q_z \forall U \exists T_p [tru\_known(Rt,T_p) \wedge (T_{pz} \geq T_p \wedge rp\_tru(Rt,Q_z,T_{pz},U) \wedge Q_z \geq 0]$.

  where:

  Rt is the ID of the tru,

  T_p is the time point at which the tru is recognised to exist,

  T_{pz} is any time point equal to or after T_p,

  U is the unit of measurement for Rt,

  Q_z is the the quantity of Rt at T_{pz}.

  This axiom addresses all developer requirements.

- A10. Constraint on rp_tru: Individual units of a tru are indistinguishable from each other, and hence traceability within a tru is not possible.

  This axiom addresses developer requirement Q3.

- A11. Constraint on rp_tru: Once 'trus are known to exist', aggregating the contents of two or more trus does not result in the aggregate quantity maintaining the ID of any of the trus that are aggregated.

  This axiom addresses developer requirement Q4.

3.4 Answering competency questions

The user competency question was this:

Q1: How much of a specific batch of resources was used by one or more activities over a given period of time?

The question can be posed in English using the terms from the ontology data model as:

Q1: What were the resource points of a given tru κ over a period of time, $[\tau_1,\tau_2]$, where the resource points are measured in υ units of measurement?

The question can be formally stated in First-order Logic using terms from the ontology as:

$Q1: \forall T_p \forall Qu(rp\_tru(\kappa, Qu, T_p, \upsilon) \wedge \tau_i \leq T_p \leq \tau_2)$

Ontology axioms are then applied to the populated enterprise model to prove this First-Order Logic theorem for the constants, κ, τ_1, τ_2, and υ. The implementation example shows the graph of resource point over time for a lamp manufacturer for which a populated enterprise model exists, for a κ = tru_bolt1_001, for τ_1 = 0 and τ_2 = 26, measured in υ = 'object' unit of measurement.

The data model of the TOVE Quality Ontology-VB is implemented in C++ using the ROCK^{TM} knowledge representation tool from Carnegie Group. The axioms are implemented in Prolog. Competency questions are implemented as Prolog queries, which are answered by applying the axioms to an instantiated data model. Queries about the enterprise model are made in Prolog. The query answers are visualised using a graphical user interface called Oak^{TM}, developed at the Enterprise Integration Laboratory.

The Prolog implementation of the competency question is shown below, along with the graphical representation of the answer[7].

Implementation:

```
show_tru_history(R,[Te,Te]):-
  rp_tru(R,Q,Te,U).

show_tru_history(R,[Ts,Te]):-
  rp_tru(R,Q,Ts,U),
                          TNew is Ts+1,
                          show_tru_history
                           (R,[TNew,Tel]).
```

Figure 4 shows that the traceable resource unit named `tru_bolt1_001` was produced by the primitive activity named `purchase_bolt1_001` at time 4 min, where the quantity of the tru was 100. Figure 4 also shows that all quantity of `tru_bolt_001` was consumed for the primitive activity named `assemble_nut_bolt1_001` at time 11 min.

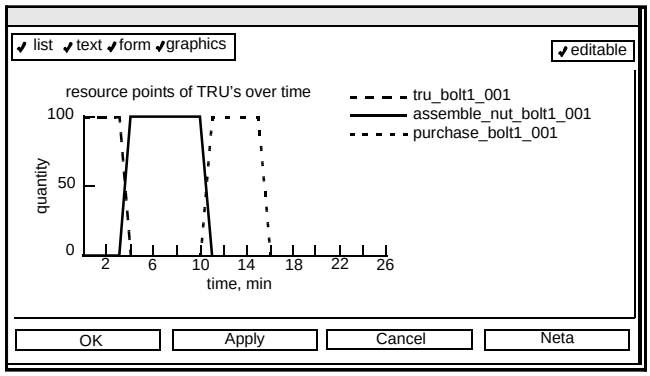


Fig 4 Graphical answer to traceability user competency question.

[7] Some of the granularity assumptions in the TOVE traceability ontology-VB are that there are integral units of time, and quantity changes occur discretely, not continuously.

4. Conclusions

The TOVE Traceability Ontology provides representations with which a quality problem identification and tracing software application can be built. Though there are many similar applications which support traceability — e.g. enterprise resource planning software like SAP [16] — most do not represent and enforce the properties of an entity which make it traceable. Traceability ontology axioms specify constraints to ensure traceability under conditions when traceability is likely to be jeopardised — when two or more similar entities are aggregated, and when one entity is split into two or more similar entities. So, not only are data models and axioms provided in the ontology to perform a trace on products (trus) and activities (primitive activities), additional axioms ensure that traceability is possible.

Kim [5] details ontology representations that extend the expressiveness of the TOVE Traceability Ontology to characterise more finely conditions for ensuring traceability; more detailed constraints on aggregating and splitting trus are provided. There are two key additional ontology extension opportunities. One is to develop representations to ensure and enable traceability for requirements; this would be crucial for a quality control application for the software development process. The other is to extend the ontology for continuous processing, say for ensuring and enabling traceability for an oil refinery.

The main evaluation criterion for the development of the TOVE Traceability Ontology-VB is competency, the capability of representations to support tasks for which it is designed. Other than discerning that axioms are consistent, detailed evaluations versus knowledge representation evaluation criteria such as completeness and closure, as well as systems performance criteria like efficiency and scalability are beyond the scope of this research endeavour. Completeness evaluations are applied for some of the other TOVE ontologies [6].

There are many augmentations to the Beta Version of the TOVE Quality Ontology. Kim [5] details a more expressive and competent set of models called the TOVE Ontologies for Quality Modelling.

To summarise, a logical formalisation of quality knowledge is presented in this paper. By formalising this body of knowledge, the following benefits have accrued:

- elucidation of the concept of quality by classification of identification, traceability, and measurement as domains explored for the development of the TOVE Quality Ontology-VB,
- clearly identified terminology and axioms for traceability — the TOVE Traceability Ontology-VB,
- presentation of a rigorous methodology for ontological engineering,
- graphical presentation of the capability to use this formalisation to make deductions and decisions about quality.

The TOVE Quality Ontology-VB is the representational basis upon which formalised quality knowledge can be used to integrate quality-related decision making throughout an enterprise.

Acknowledgments

This research is supported, in part, by the Natural Science and Engineering Research Council, Digital Equipment Corp, Micro Electronics and Computer Research Corp, and Spar Aerospace.

References

1 Hauser J R and Clausing D: 'The House of Quality', Harvard Business Review, pp 63—73 (May-June 1988).

2 Fox M S, Chionglo J C and Fadel F G: 'A common-sense model of the enterprise', in 2nd Industrial Engineering Research Conference Proceedings, Los Angeles, CA (May 1993).

3 Godfrey A B: 'Ten clear trends for the next ten years', in Quality Quotes, 19, No 2 (Spring 1993).

4 Brachman R J: 'On the epistemological status of semantic networks', in Findler N V (Ed): 'Associative Networks: Representations and Use of Knowledge by Computers', Academic Press, pp 3—50 (1979).

5 Kim H M: 'Representing and reasoning about quality using enterprise models', PhD Thesis, Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Ontario, Canada (1999).

6 Grüninger M and Fox M S: 'Methodology for the design and evaluation of ontologies', Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95, Montreal (1995).

7 Fadel F G, Fox M S and Grüninger M: 'A generic enterprise resource ontology', in Proceedings of Third Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, Morgantown, WV, pp 117—128 (April 1994).

8 Fox M S, Barbuceanu M, and Grüninger M: 'An organisation ontology for enterprise modelling: preliminary concepts for linking structure and behaviour', in Fourth Workshop on Enabling Technologies — Infrastructures for Collaborative Enterprises (WET-ICE 95), West Virginia University, pp 71—81 (April 1995).

9 ISO General Secretariat: 'ISO 9000 International Standards for Quality Management', Geneva, Switzerland (1991).

10 Garvin D A: 'What does 'Product Quality' really mean?', Sloan Management Review (Autumn 1984).

11 Crosby P B: 'Quality is Free: The Art of Making Quality Certain', McGraw-Hill, New York (1988).

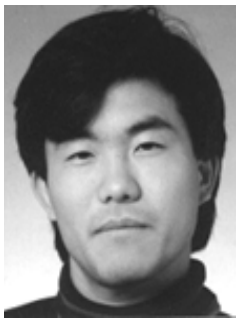
12 Boehm B: 'Software engineering economics', Prentice Hall (1981).

13 Di Franca G T: 'The investigation of the physical world', Cambridge, Cambridge University Press, England (1981).

14 Gruber T R: 'Towards principles for the design of ontologies used for knowledge sharing', Technical Report KSL 93-4, Knowledge Systems Laboratory, Computer Science Department, Stanford University, Stanford, CA (1993).

15 Grady J O: 'System requirements analysis', McGraw-Hill Inc (1993).

16 SAP AG: 'SAP Solutions', — http://www.sap.com: (August 1999).

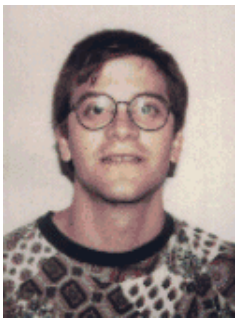


Henry Kim is an Assistant Professor of Information Systems at the Schulich School of Business, York University in Toronto, Canada. He received his PhD in Industrial Engineering from the University of Toronto with a thesis titled, "Representing and Reasoning about Qaulity using Enterprise Models'.

He is interested in ontology-based information systems that facilitate knowledge sharing by using shared vocabulary and business rules. He is also interested in applying these systems for enterprise modelling, knowledge management, systems integration, and eCommerce.

He was a Short-Term Fellow at BT Laboratories in the Intelligent Business Systems Research Group, conducting research into using ontologies to integrate data from heterogenous BT systems. He has also worked in industry as a management consultant, technical consultant, software engineer, and industrial engineer.

Mark Fox received his BSc in Computer Science from the University of Toronto in 1975 and his PhD in Computer Science from Carnegie Mellon University in 1983. In 1979 he joined the Robotics Institute of Carnegie Mellon University as a Research Scientist. In 1980 he was appointed Director of the Intelligent Systems Laboratory. He co-founded Carnegie Group Inc in 1984, a software company which specialises in knowledge-based systems for solving engineering, manufacturing, and telecommunications problems. Carnegie Mellon University appointed him Associate Professor of Computer Science and Robotics in 1987 (with tenure in 1991). In 1998 he was appointed Director of the CMU Center for Integrated Manufacturing Decision Systems. In 1991, he returned to the University of Toronto where he was appointed the NSERC Research Chairholder in Enterprise Integration and was appointed Professor of Industrial Engineering, Computer Science and Management Science. In 1992, he was appointed Director of the Collaborative Program in Integrated Manufacturing. In 1993, he co-founded Novator Systems Ltd, a company that provides E-Retail services over the Internet.



Michael Grüninger has been a Research Scientist in the Enterprise Integration Laboratory at the University of Toronto since 1993 and is Project Manager of the Enterprise Engineering Project.

He received a BSc in Computer Science from the University of Alberta in 1987 and his MSc in Computer Science from the University of Toronto in 1989. His doctoral work at the University of Toronto has been in the area of logic and object recognition in computer vision, constructing ontologies to support 2-D object recognition in scenes with occlusion.

He currently supervises the development of TOVE (Toronto Virtual Enterprise) within the Enterprise Integration Laboratory.