Proposed
Draft Standard for Learning Technology—
Simple Reusable Competency Map

DOCUMENT STATUS: INCOMPLETE DRAFT IN PROGRESS, SUBJECT TO RADICAL CHANGE.
THIS ROUGH DRAFT IS CIRCULATED FOR FEEDBACK, COMMENTS, CRITIQUE AND IMPROVEMENTS.
CO-AUTHORS ARE WELCOME – CONTACT CLAUDE AT OSTYN.COM

This document is a draft for a Standards project to be submitted to the Learning Technology Standards Committee of the IEEE Computer Society

Copyright © 2006 by Claude Ostyn

All rights reserved. This document is an unapproved draft of a document intended for submission as a possible IEEE Standard and is subject to change without notice. Permission is hereby granted for Ostyn Consulting customers and partners, and for participants in industry Standard initiatives, to reproduce and redistribute this document for purposes of Standardization services. If this document is to be submitted to any Standards body or included in such a submission, notification shall be given to Claude Ostyn. Other entities seeking permission to reproduce all or any portion of this document must contact Ostyn Consulting at www.ostyn.com for the appropriate license.

Abstract

This Standard defines an information model for simple, reusable competency maps to be used for describing, referencing, and exchanging data about the relationships between competencies, primarily in the context of online and distributed learning. The information model allows the representation of relationships between competencies or complementary aspects of competency, such as they have often been captured in competency models. In this Standard, the word competency is used in a very general sense that includes skills, knowledge, attitude, and learning outcomes. This Standard references the IEEE P1484.20.1 Reusable Competency Definition (RCD) Standard (in draft status at of this writing). The RCD Standard specifies how any arbitrary competency definition can be captured using Standard data model. In this Standard, the relationships between RCDs are represented is a map in which each node may reference a RCD. Simple "roll up" rules are specified to allow representation of how sub-competencies specified in the map can "roll up" to a broader competency.

Keywords:

competency, competency definition, objective, competency map, competency taxonomy, hierarchy, directed acyclic graph

[NOTE: Information about IEEE LTSC P1484.20 can be found at:

`http://www.ieeeltsc.org/wg20Comp/`

This note will be removed upon reaching the final draft of this IEEE document.]

Introduction

(This introduction is not a part of xxxx, Draft Standard for Learning Technology—Data Model for Reusable Competency Map.)

This Standard defines a data model for describing, referencing, and exchanging simple lists or hierarchical maps representing the relationships between reusable competency definitions, primarily in the context of online and distributed learning.

Participants

At the time this Standard was completed, the working group had the following membership:

TBD *Chair*		
(other names to be updated)		

The following persons were on the balloting committee: (To be provided by Standard editor at time of publication.)

Acknowledgements

TBD.

Contents

Draft Standard for Learning Technology—Simple Reusable Competency Map

1 Overview

This Standard defines an information model for simple, reusable competency maps to be used for describing, referencing, and exchanging data about the relationships between competencies, primarily in the context of online and distributed learning. The information model allows the representation of relationships between competencies or complementary aspects of competency, such as they have often been captured in competency models. In this Standard, the word competency is used in a very general sense that includes skills, knowledge, attitude, and learning outcomes. This Standard references the IEEE P1484.20.1 Reusable Competency Definition (RCD) Standard (IEEE project in draft status at of this writing). The RCD Standard specifies how any arbitrary competency definition can be captured using Standard data model. In this Standard, the relationships between RCDs are represented as a map in which each node may reference a RCD. Simple "roll up" rules are specified to allow representation of how subcompetencies specified in the map can "roll up" to a broader competency.

The Standard specifies that the must be a directed acyclic graph. A related Standards project may specify other map shapes, such as a general directed graph, in order to allow the representation of competency models of arbitrary complexity, but such graph shapes and representations are outside the scope of this Standard. This Standard enables interoperability among learning systems and competency management systems that deal with competency information by providing a means for them to refer to common competency definitions and to model the relationships between those common definitions through the use of interoperable maps. This Standard is not intended to define ontologies or to duplicate semantic mapping Standards such as topic maps; however, it allows the representation of competency maps as directed acyclic graphs generated by processes that traverse ontologies or topic maps.

Different communities of practice tend to use different models to represent the competencies required for tasks, readiness for a job, or mastery of a subject domain, but increasingly they often need to exchange and manage competency information with each other. The information model in this Standard can be used to exchange these models between learning systems, human resource systems, learning content, competency or skills repositories, and other relevant systems. The SRCM that conform to this Standard are intended for interchange by machines, just like the RCD that conform to the RCD Standard.

1.1 Scope

This Standard shall specify the mandatory and optional data elements that constitute a Simple Reusable Competency Map as used in a Learning Management System, or referenced in a Competency Profile. This Standard is intended to satisfy the following objectives:

— Provide a Standardized data model to represent the parent, child or sibling relationships that may exist between Reusable Competency Definitions.

— Reconcile various existing and emerging data models into a widely acceptable model

— Provide a Standardized way to represent the result of grouping and decomposing competency information captured in Reusable Competency Definitions. Nodes that represent groupings or products of decompostion can be mapped to new or existing Reusable Competency Definitions.

— Provide a unique identifier as the means to unambiguously reference a Simple Reusable Competency Map, regardless of the setting in which this Competency Map is stored, found, retrieved, or used. For example, metadata that describe learning content may contain a reference to one or more Simple Reusable Competency Maps that provide the context for a classification of the learning objectives for the content.

— Provide a Standardized data model for additional information about a Simple Reusable Competency Map, such as a title, description, and source, compatible with other emerging learning asset metadata Standards

— Given any Reusable Competency Definition and a Simple Reusable Competency Map that contains a reference to this Reusable Competency Definition, provide the data structure that allows the discovery of related competency definitions through the structure of the map.

— Provide a Standardized data model for additional information about relationships between groups of competencies represented by a Reusable Competency Map Definition, such as relative weights, rollup rules governing how component competencies can be considered to add up to higher level competencies, and proficiency levels required to assert mastery of a competency in the context of a hierarchy of competencies defined by the map.

— Provide a controlled vocabulary to express how Simple Reusable Competency Maps are semantically related in a list or hierarchical model.

This Standard specifically does not cover:

— A data format, bindings or coding, except as minimally required for the purpose of exchange between compliant implementations

— Quality and accuracy in the data itself, although it will describe recommended best practices. For example, this Standard does not cover the quality or validation of the various component competencies that make up a higher level competency, or the relevance of the Reusable Competency Definitions referenced by a node in a Reusable Competency Map.

— The processes by which a SRCM is created, generated, maintained or published.

— How the relationships between competencies are stored in a database or learning management system.

— Any representation of relationships between competencies that requires a general graph rather than a list or directed acyclic graph.

— Mapping or references to data objects other than Reusable Competency Definitions or other Simple Reusable Competency Maps.

— Certification data models and how they may be referencing SRCMs

[Note (to be moved somewhere else). However, Certification records may reference Simple Reusable Competency Maps along with Competency Definitions. For example, an accredited authority may grant certificates that acknowledge that an individual meets the requirements for a particular competency after walking an associated Simple Reusable Competency Map to identify requirements for component competencies that add up to the target competency.]

— Individual competency records, as would be found in the competency profiles of individuals or groups, and how they may be referencing SRCMs.

[Note (to be moved somewhere else) However, such records can include references to specific Simple Reusable Competency Maps along with references to Reusable Competency Definitions. For example, a competency profile for an individual may include a collection of certificates which in turn reference Competency Definitions, and that collection may be discovered by inspecting a Simple Reusable Competency Map that references any of those Competency Definitions. Skill gaps may be discovered by finding the map nodes for which no evidence of competency exists.]

— Confidence or trustworthiness of competency records; confidence in results of summation that includes rolling up proficiency for multiple competencies shown by a map to be related, where some competency records may vary in trustworthiness..

1.2 Purpose

The purpose of this Standard is to define a universally acceptable Simple Reusable Competency Map Definition model to allow the creation, exchange and reuse of hierarchical maps representing the relationships between Competency Definition in applications such as Learning Management Systems, Competency or Skill Gap Analysis, Learner and other Competency profiles, and so on. The Standard is needed because there are currently many definitions of the terms “Competency Model”, "Competency taxonomy", "Learning Objective", "Competency" and "Skill", and very little agreement between how those definitions can be used to define reusable data models to support automation and computer-assisted discovery. This Standard uses a general definition that can be applied to any scale of competency hierarchy, from shallow ad-hoc taxonomies used in an assessment to deep formal hierarchies representing a do-

main, while conserving the same data model regardless of how strictly a particular organization or institution requires the data to be formulated. This Standard also addresses the following needs:

— A common data model that allows the building of various ad hoc or formal models, inventories, hierarchies and mappings of Competency Definitions.

— A Standard that allows persistent, long lived Simple Reusable Competency Maps to be created, exchanged among systems, and maintained.

— A Standard data element by which Simple Reusable Competency Maps can be identified as globally unique among compliant systems and repositories.

— A common data model to represent the result of aggregating or decomposing competency information captured in RCDs through the use of additional RCDs representing the products of the decomposition, and to capture assertions about the hierarchical relationship between the original RCD and the subsidiary RCDs in the form of a Simple Reusable Competency Map.

— A common data model for the descriptive or cataloging metadata that give a reusable Simple Reusable Competency Map its value in a reuse environment. Such metadata may typically include the publisher of the Simple Reusable Competency Map, validation information, and other descriptive information useful to locate an existing competency map in a repository catalog or collection index.

— Correspondence with the IEEE Learning Objects Metadata Standard (IEEE 1484.12.1) developed by a parallel group.

2 References

The following referenced documents are indispensable for the application of this Standard. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

(P)1484.20.1 Standard for Learning Technology – Reusable Competency Definitions

(TBD: 1484.12.1 LOM)

3 Definitions

For purposes of this Standard, the following terms and definitions apply. IEEE 100, *The Authoritative Dictionary of IEEE Standards Terms*, Seventh Edition [A3], should be referenced for terms not defined in this Clause.

(TBD)

LangString bag: A structured datatype that represents one or more character strings. A LangString may include multiple semantically equivalent character strings that represent translations of the same meaning into different languages. See also: datatype.

competency: For this Standard, a competency is defined as any form of knowledge, skill, attitude, ability or learning outcome that can be described in a context of learning, education or training.
Note—The word competency here is to be interpreted in the most broad sense to include educational objectives (those things that are sought) and competency or competencies (those things that are achieved). The word "competency" is also used to include all classes of things that someone, or potentially something, can be competent in, although some communities of practice use the word with nuance, for example limiting its use to skill and excluding knowledge or understanding. In learning contexts, a competency to be achieved is often named "objective". Competency information represented in a competency map may also include facets of competency, such as cognitive, psychomotor and affective, or behavioral indicators of competency.

smallest permitted maximum: For implementation-defined values, the smallest permitted maximum value. See also: clause 4.5.

value space: The set of values for a given datatype (ISO/IEC 11404:1996).

NOTE:--In this Standard, a value space is typically enumerated outright, or defined by reference to another Standard or specification.

3.1 Abbreviations and acronyms

[Editing note: Need to order and remove acronyms not in Standard]

ADL Advanced Distributed Learning

IMS IMS Global Learning Consortium

ISO International Standards Organization

JTC Joint Technical Committee

LTSC Learning Technology Standards Committee

RCD Reusable Competency Definition

RDCEO IMS Reusable Definition of Competency or Educational Objective

SCORM Shareable Courseware Object Reference Model

W3C World Wide Web Consortium

XML Extensible Mark-up Language

SPM smallest permitted maximum

URI: Uniform Resource Identifier

URN: Uniform Resource Name

4 Conformance

Conformance to this Standard is discussed in 4.1 – 4.5.

In this Standard, "shall" is to be interpreted as a requirement on an implementation; "shall not" is to be interpreted as a prohibition.

Note. Since this Standard defines a data model but not a specific binding, and system conformance cannot be defined without one or more binding, conformance of systems is outside the scope of this Standard.

4.1 Data instances

A conforming data instance shall be an instance of the data model as defined in Clause 6.1.

4.2 Smallest permitted maximum values

In this Standard, smallest permitted maximum values are defined for:

— Items with multiple values: All applications that process SRCM instances shall process at least that number of entries stated. In other words: an application may impose a maximum on the number of entries it processes for a data element with multiple values, but that maximum shall not be lower than the smallest permitted maximum value.
— Data elements with type CharacterString or LangString: All applications that process SRCM instances shall process at least that length for the CharacterString value (either directly or contained in the LangString) of that data element. In other words: an application may impose a maximum on the number of characters it processes for the CharacterString value of that data element, but that maximum shall not be lower than the smallest permitted maximum value for the data type of the data element.

This Standard defines smallest permitted maximum (SPM) values for data elements with data types that include bag, set, and characterstring. For these data elements, an implementation that conforms to this Standard shall accept and process at least that number of entries or characters specified by the SPM for the element and may accept and process a larger number.

NOTES:

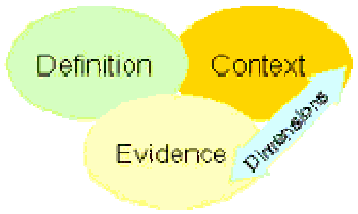
1—The intent is for the SPM values to cover most cases.

2—What "processing" means in the above depends on the nature of the application.

3—This Standard does not define any provision for how and whether a system can process more than the SPM for a particular data element.

5 Conceptual model (informative)

5.1 Functional overview



Competency data may include

- Reusable (generic) **definition** of the competency
- **Evidence** of competency
- **Context** within which the competency is defined, or that defines the competency
- **Dimensions** such as proficiency on a scale or time

Figure 1 —Reusable competency definitions capture only a part of the data that define a competency

Reusable Competency Definitions (RCD), as defined in IEEE P1484.20.x, may used to capture a competency definition at any level of specificity, from the most precise to the most general. The more specific a RCD is, the less reusable it is. Often a less precise definition is very useful, especially when trying to compare competency data between different communities of practice. For example, as a tourist you are typically considered competent to drive a car in a foreign country, even though the details of the competency model for driving competency may be very different between countries. In that case, the gross competency definition is good enough and going into details would impair commerce when it comes to rending a car. In other cases, though, you do want to be able to reference a competency in the context of a specific model that corresponds to the expectations or requirements of a specific community of practice. Such a model can be represented by a competency map.

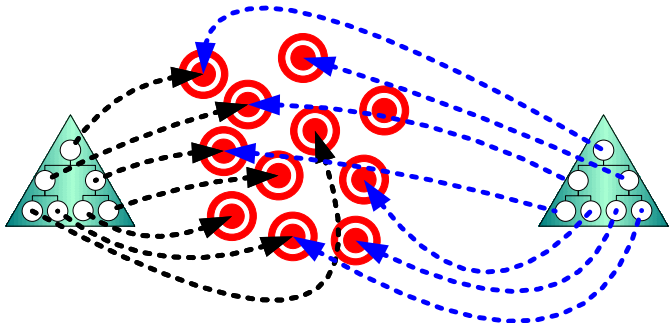


Figure 2 — Competency maps may represent different models of the same competency

5.1.1 Modeling

The SRCM data model is minimalist and extensible. It is purposely neutral with regard to models of competencies and the use of competencies. Competencies and competency models are defined and structured in many ways in different communities of practice. This Standard allows many communities of practice to exchange useful information regardless of the model they use, as long as the model can be represented or exported in a shape that can be represented according the Standard.

5.1.2 Directed acyclic graph (DAG) map shape

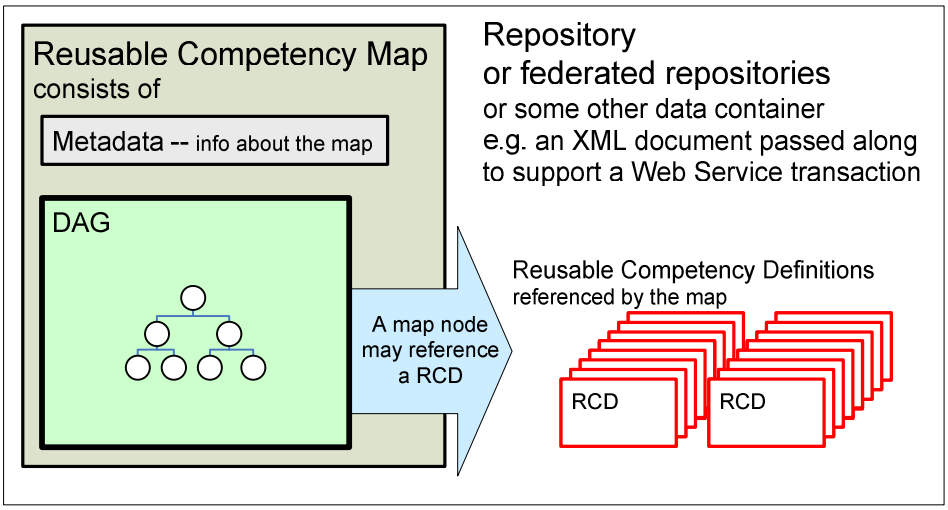


Figure 3 — Competency maps as a graph with references to RCDs

The Standard specifies the shape of a SRCM as a directed acyclic graph (DAG). A DAG is a hierarchical collection of nodes that implies containment. If node A has children B and C, it implies that A contains B or C. In competency terms, this would imply that competency A may be decomposed into sub-competencies or competency facets B and C, or that B and C contribute to A. In reality, many so-called existing competency "taxonomies" are not true taxonomies, because the same competency components tend to appear more than once in the

model hierarchy. The Standard allows this to be represented in a DAG. The DAG allows a node to have more than one parent, as long as the parent cannot be a descendant of the node. Figure 4 shows four examples of DAGs, from simple to more complex. In the figure, arrows represent a parent-child relationship, with the arrows going from parent to child.

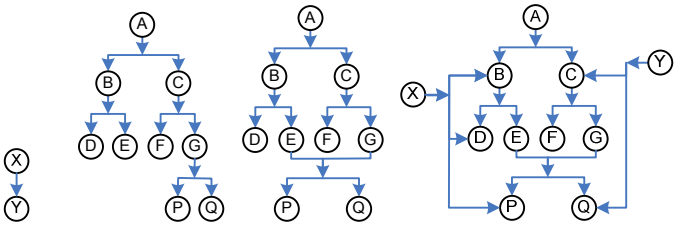


Figure 4 — Some DAG topologies with different levels of complexity.

A simple topology for a DAG is a tree with a single root node and no commonality between sub-trees within the tree. A more complex topology may specify common children for more than one node, or more than one origin node. For example, in the rightmost graph in Figure 4, A, X and Y represent different competencies that have certain component competencies in common.

Nodes in a map can have specific rules useful for modeling. Different rules can apply to individual nodes.

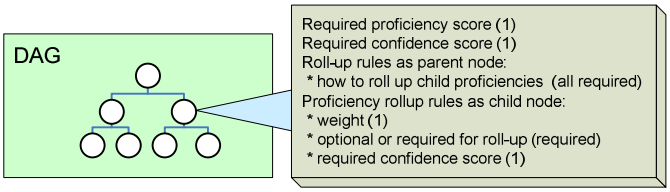


Figure 5 — Rules may be associated with nodes to define interpretations of the map

The fact that a rule is defined in the Standard does not imply that it is required in all applications. For example, the required score required rules in Figure 5 may be required in an application profile for an assessment request, but may not be important when reporting a competency inventory for a person. In any case, the Standard defines a default value for each rule. For example, by default it is assumed that proficiency is required for all "children" of a defined competency in order to achieve proficiency for the "parent".

An application of the competency map may add other rules, but this is outside the scope of this Standard. For example, a competency map may be used to specify how to roll up proficiency data throughout a collection of competency records that reference the RCDs that are represented in the map. Each competency record typically references a RCD and includes a data element for proficiency status and maybe proficiency score. An application that builds or maintains a monolithic personal competency profile, rather than using cross references, might merge data from those competency records with the structure of the competency map, in which case the resulting data structure would include not only the nodes of the competency map, but also data elements such as proficiency status for each node.

5.1.3 Map extension by reference

Maps may be symbolically merged by references to other maps. Any node in a map may reference a RCD, another map, or both. This is useful to allow the association of a more detailed map with a RCD. For example, a list of competency definitions may be extended by referencing maps that provide detailed breakdowns the components of those competency definitions as assumed by the creator of the list.

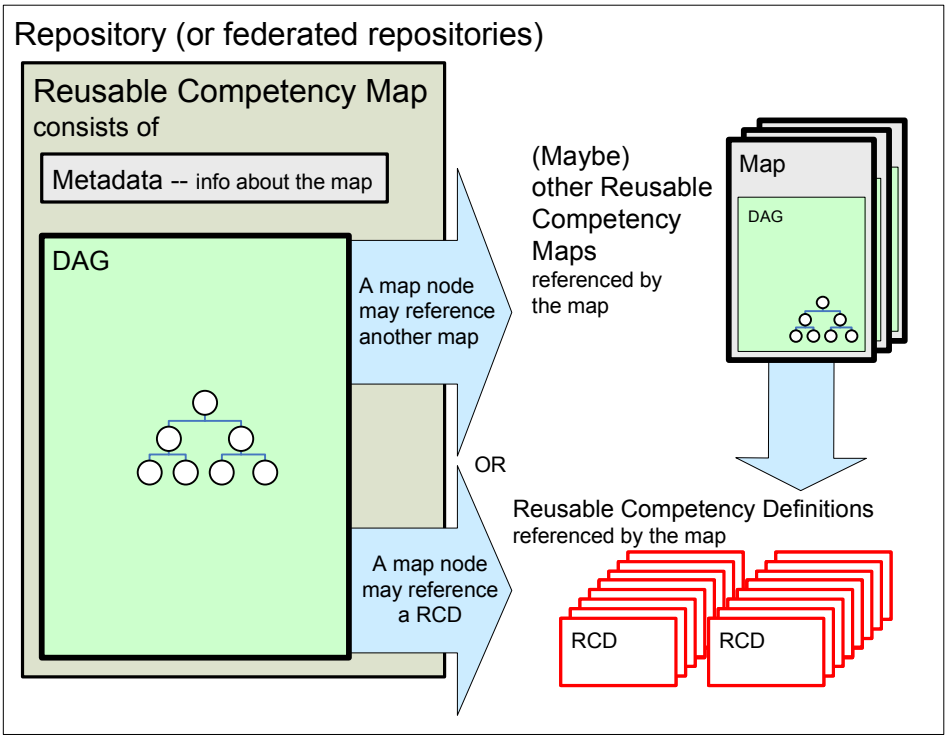


Figure 6 — Competency maps may also include other maps by reference

Application profiles may put constraints on the shapes that can be merged because of the complexity of the resulting map. Application profiles may also specify that when maps are merged, the referencing node is treated as a functional boundary, and that references to other maps are followed only on demand in a more costly operation.

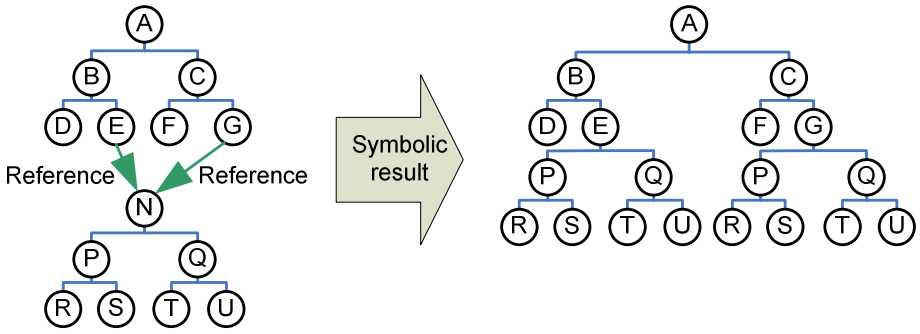


Figure 6 — Extension of a DAG map by reference to another DAG map

5.1.3.1 Map extension rules and constraints

If a node in a DAG shaped map references another DAG shape map with a single origin ("root") node, the root node of the DAG is subsumed in the referencing node. A node in a DAG competency map can therefore represent any other DAG competency map. Several constrains do apply in order to retain sanity:

-- The result cannot be a cyclical topology. In other words, a node cannot become a descendant of itself as a result of merging graphs, because this violates the basic acyclic constraint of a DAG. Implementations must treat any node that would violate this constraint as a leaf node and ignore its reference to another map. For example, an implementation may use the same method used by a web site spider to avoid visiting pages or directories that have already been visited through another path.

-- The referencing node retains its properties as child node, but the properties of the referenced node in its role as parent remain intact. In the example in Figure 6, nodes E and G must use the rollup rules defined by N for its children. If N references a RCD, and E also references a RCD, the RCDs are considered equivalent. In this case, when E is inspected in the context of its parent, i.e. when E is playing the role of child, the RCD that is visible is the RCD referenced by E. However, when E is inspected in the context of its children, the RCD that is visible is the RCD referenced by N, because E in its role as parent inherits the properties of N.

-- If the target of a reference is a DAG with more than one origin, the reference must specify which origin to use. Otherwise it cannot be resolved. This adds some complication to implementations. For this reason, application profiles may want to restrict implementations to single-origin DAG topologies.

5.1.3.2 Using a DAG map to represent equivalencies and similarities

A known and expected problem with the RCD model is that different communities may define equivalent or similar RCDs because they are unaware of each other's work. This becomes a problem when competency models or records must be reconciled. This can happen, for example, when two companies merge or when educational Standards are consolidated across jurisdictions. Since SRCMs can be used to represent relationships between RCDs, a natural use of SRCMs is to represent simple equivalencies and similarities. For example, the SRCM that defines such a competency equivalency can be referenced as evidence to justify the updating of a competency record that references the equivalent competency.

Figure 7 — A SRCM representing a one-way equivalency.

Figure 8 shows how a straightforward equivalency can be represented in a SRCM. Figure 7 should be read as follows: Y is the only child of X, therefore proficiency in Y implies proficiency in X. Note that the reverse equivalency, if it is true, cannot be represented in the same SRCM because that would violate the acyclical constraint of a DAG. In practice, this is not a

problem, because competency equivalencies are often not reversible. For example, a country may accept diplomas from another country, but the reverse may not be true without some additional qualifications, as illustrated in Figure 8.

Figure 8 should be read as "Y has children X and Z. Therefore proficiency in Y requires proficiency in X as well as proficiency in Z".

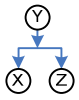


Figure 8— A SRCM representing a qualified reverse equivalency

A common problem in dealing with competency modeling is imperfect equivalencies that are not reciprocal, as seen above for X and Y. The equivalency assumptions between competencies X and Y can however be expressed with two maps, as shown in Figure 9. Figure 9 should be read as follows: "Two SRCMs represent the relationships between competencies X and Y. Proficiency in Y implies proficiency in X. Proficiency in X implies proficiency in Y only if there is also proficiency in Z."

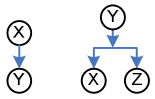


Figure 9—Set of SRCMs representing the equivalency between competencies X and Y

Note that the graphs in Figure 9 may not be merged into a single DAG, since that would violate DAG topology rules. It is the set of graphs that provides the semantics for the imperfect equivalency. In a practical application, it is likely that only one of the graphs would be used, since the available data would determine which node to use as a starting point to navigate the graph in a single direction. For example, in an operation to find which competency might encompass X, the graph on the left would be rejected immediately since there is no parent for X, but the graph on the right can be navigated from X to find Y as an ancestor.

5.1.4 Data capture

The data in a SRCM can come from many sources. Typical existing sources include various competency models developed by government agencies, academic institutions, enterprise and military training departments.

Many existing models are hierarchies or taxonomies that can be captured in more or less automatic ways, depending on how they are encoded. The process might involve splitting the data from the model into two data models – RCD (definitions) and SRCM (how the definitions are related). A useful refinement on the process might be to look up existing RCD repositories to see whether suitable RCDs have already been defined. The process would then use those RCD rather than creating new ones when possible.

The United States O*NET database can be mined for specific competency maps corresponding to a job description that someone builds by walking through the Standard occupational descriptions and ability weights captured in O*NET. For example, for a job that involves accounting and management of a loading doc as well as supervision of twenty employees, an HR person with the appropriate helper tool could identify the appropriate Standard occupation codes in O*NET and automatically generate a map of the required competencies and relative importance for the required position, tweak it for the specific requirements of the enterprise, then publish the map or pass it on to recruiting agencies.

A cursory search of the World Wide Web turns up many hierarchical competency models with considerable overlaps, many of which can be remapped easily into a SRCM instance with associated RCDs. For example, an inventory of such models for the domain of military leadership has been collected in the USA by the Air War College Center for Strategic Studies -- http://leadership.au.af.mil/sls-skil.htm#models [Ed. Note: this to be fixed up as a proper reference]

Standards based technical documents such as S1000D conformant instances can also be mined for the structure and content of a competency map. For example, a S1000D compliant technical manual for the maintenance and operation of a vehicle could be used as the source for competency maps segmented for maintenance personnel and vehicle operators. In this example, competency maps could also be updated automatically or semi-automatically as the described equipment evolves. The application that extracts this information from the S1000D instance could automatically guide the generation of new reusable competency definitions and the creation or retirement of competency map nodes corresponding to feature additions or removals in the vehicle's documentation.

5.1.5 Example applications

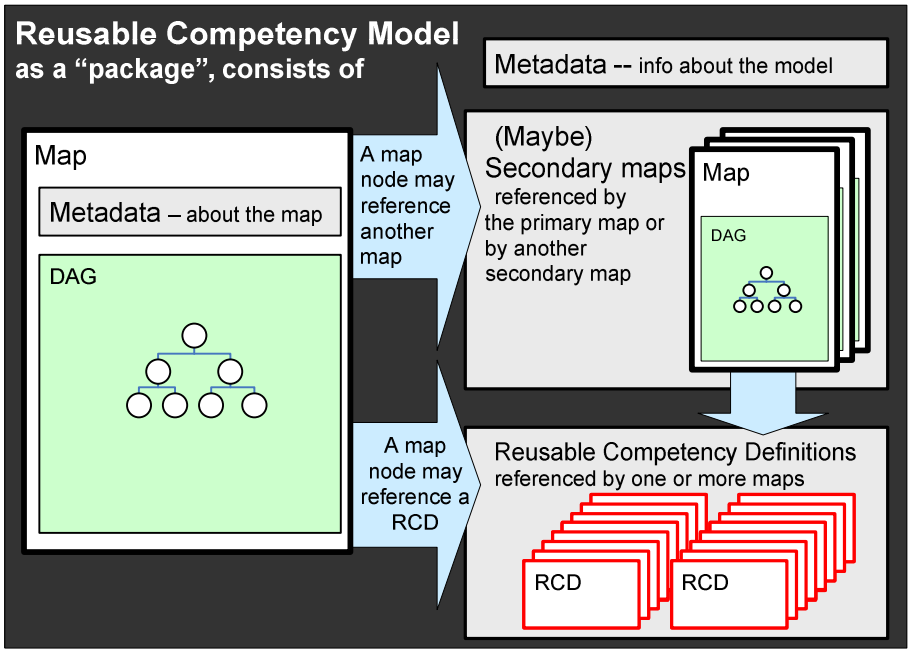


Figure 10— Application example: Packaging a job competency profile

A competency map can be used to represent the relationships between competency definitions in:

- a job competency profile (requirements for the job)

- a personal competency profile (acquired competencies)

- a personal competency gap profile (competencies to learn)

Some other existing or emerging competency data collection specifications or practices may not flexible enough to represent all models, or do not support any meaningful interoperability. This proposed map Standard, along with related Standards, would allow the capture of key competency assumptions into a Standard, interoperable way. The competency information could be captured by translating from proprietary or portfolio to a competency map with associated RCDs and possibly competency evidence records if that is part of the original data.

An application of this would be to allow look up of corresponding nodes in a map detailing the specific sub-competencies and their relative importance for a job requirement, with the RCD as a key (including the recording of equivalencies or dependencies discovered in ontologies, for example) Once a corresponding node is discovered, its relation links can be explored to guide further analysis. This analysis does not have to be automated to be useful--it could be presented as a guide to a human examiner. For example, if the match is on something like "English verbal skills", the examiner could be presented with an outline of the sub-skills required for the job and compare that with the outline of sub-skills derived from the map that results form the capture of the applicant's portfolio into a competency map.

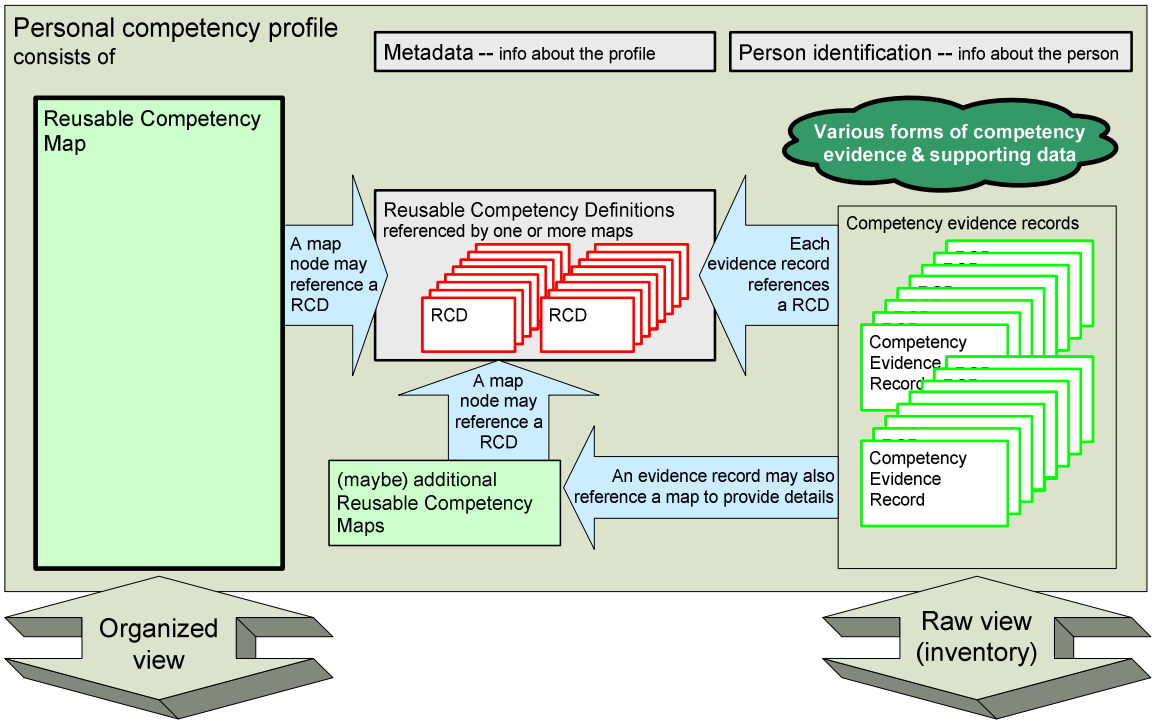


Figure 11 — Application example: A personal competency profile

Another example application would be in connection with learning object metadata, as shown in Figure 13. The competency taught by a learning object, or tested by an assessment object, should be represented by a reference to a reusable competency definition in the metadata. In IEEE LOM, this would be defined in one or more Classification elements. Someone looking for learning resources for a particular competency could then locate any competency map that contains a reference to the corresponding reusable competency definition, and then use that map to find higher level, related competency definitions. Searching for learning objects that specify those reusable competency definitions may then find learning objects that contain relevant resources, even when a one to one match on the original competency definition might fail. This can be made more reliable if the competency model assumed by the designers of the learning resource is made available in the metadata by referencing a reusable competency map.

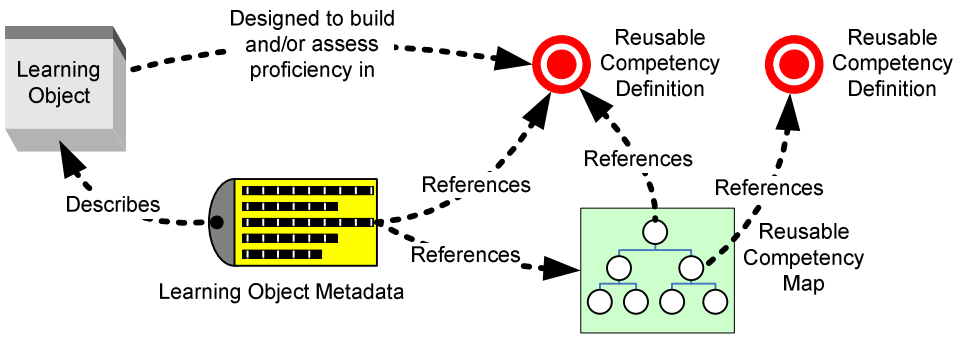


Figure 12 — Application example: Finding related learning resources

5.2 Data model overview

The data model contains the following mandatory elements:

a) Identifier: A globally unique label that identifies this Reusable Competency Map. This identifier uses the same data elements as the Identifier element defined in the IEEE LOM Standard, and consists of two sub-elements: Catalog and Entry. The Identifier is sufficient to reference the competency in any other system. The Identifier may be a handle or digital object identifier according to emerging Standards and practice, e.g. the Handle system and the CORDRA specification.

b) Title: A text label for the Reusable Competency Map. This is a short human-readable name for the taxonomy. While the Identifier provides the definitive reference to the definition, it is typically unintelligible. The Title provides a convenient alternative readable form, but one which is not the definitive label. The Title may be repeated in multiple languages.

The other elements defined by the data model are optional:

c) Description: A human readable description of the map. This is an optional unstructured (opaque) “text blob” meant to be interpretable only by humans. The Description may be repeated in multiple languages.

d) Other information: Other information about the map, including optional metadata.

d) Graph: A structured collection of nodes that represent the hierarchical relationship between competencies. There is always at least one node. Some or all the nodes may reference a Reusable Competency Definition, which is not part of the data model, or another map that is outside the scope of the map that contains the node. This reference is through an identifier with global scope and does not specify a particular location, but rather the identifier of the RCD or map, wherever it may be found. A node does not have to reference a RCD or map. This may be because the node exists only for the purpose of grouping, because an appropriate RCD or map has not yet been found or created for reference, or because a reference to a RCD or map was found to be invalid and has been removed.

A node has a Title, which is a human readable name for the node. This Title may be repeated in multiple languages. If no Title is defined for a node, but the node references a Reusable Competency Definition or map, the Title may be obtained automatically from that Competency Definition or map. An implementation may also obtain additional data from the referenced object by looking up the referenced object.

A node may have associated rules that specify how implementations may aggregate summary information about mastery of the referenced competencies. For example, one set of rules applies when the node is considered as a parent in the hierarchy, and another set applies when the node is considered as a child in the hierarchy. In the absence of explicit rules specified by the creator of the map, implicit default rules are specified by this Standard.

This Standard does not define a specific extension mechanism for the data model. Implementers may create additional data models for competency data and the representation or encoding of relationships between competencies. Such models may be used to augment this model to support different communities of practice.

(End of informative clauses)

6 Data model

This Clause defines the data elements of a SRCM

Unless noted otherwise, all components of "records" are optional in a data instance.

NOTES:

1—The use of ISO/IEC 11404 notation in the synopses in 6.1 and 6.2 is for descriptive purposes only. A complete implementation of the operations defined in ISO/IEC 11404 is not required for conformance.

2—The ISO/IEC 11404 notation describes the semantics of the language-independent data types across all bindings (e.g., implementation of a data type as itself, its subtypes, its subclasses, and its specializations). For example, an ISO/IEC 11404 "record" may be implemented as an SQL table row, or as an XML complexType; an ISO/IEC 11404 "characterstring" may be implemented in an encoding (ISO 646, ASCII, ISO 8859-1, UTF-8, UTF-16, UTF-32, etc.) that supports the repertoire specified in the parameter to characterstring data type.

3—All examples in 6.1 and 6.2 are informative and do not endorse any particular binding.

4—The following language-independent data types used in this Standard are defined in ISO/IEC 11404: bag, characterstring, record, set, state.

5—The labels for data elements and data types in the synopses in clauses 6.1 and 6.2 are for reference only. There is no requirement that an implementation use the exact same labels, as long as the data elements and data types are semantically equivalent.

6—This Standard does not define a specific extension mechanism for the data model. Implementers may define binding that allow additional elements, or create additional data models for competency data. Such models may be used to augment this model to support different communities of practice.

6.1 Reusable Competency Mape

Synopsis

```
reusableCompetencyMap :
record
(
   identifier :  longIdentifierType,
```

```
      rcdRef : longIdentifierType,
            // optional
      title : langStringType(1000),
            // the parameter value is the SPM
            // optional
      description : langStringType(4000),
            // the parameter value is the SPM
            // optional
      srcmSchemaLabel: characterstring(iso-10646-1),
                  // SPM: 1000 characters
      srcmSchemaVersion: characterstring(iso-10646-1),
                  // SPM: 1000 characters
      referential: boolean,      metadata :      bag of anyType,
                  // SPM: 10 of any type in the bag
      graph : ( see clause 6.1.8),
      extensions : anyType
   )
```

Description

The components of `reusableCompetencyMap` are defined in 6.1.1 – 6.1.9. Depending on the implementation, an instance of `reusableCompetencyMap` shall include one or more of the defined components. The map element is a required component. It may not be empty.

All top level elements in the `reusableCompetencyMap` data model are intrinsically unordered.

Note—A binding may impose a particular ordering on SRCM data elements that conform to that binding. Other than conformance to the binding, no significance is associated or should be inferred from that ordering requirement.

6.1.1 Identifier

Synopsis

```
   identifier :
         longIdentifierType,
```

Description

A globally unique label that identifies this Reusable Competency Map. The Identifier is sufficient to reference the competency taxonomy in any other system.

Subclause 6.2.1 defines longIdentifierType.

NOTE— This identifier uses the same data elements as the Identifier element defined in the IEEE LOM Standard, and consists of two sub-elements: Catalogue and Entry.

6.1.2 Title

Synopsis

```
title : bag of langStringType(1000),
   // SPM: 20 instance of langStringType in the bag
   // The parameter value is the SPM for the langString
```

Description

A single mandatory text label for this SRCM. This is a short human-readable name for the taxonomy.

Subclause 6.3.1 defines langStringType.

NOTES:

1—The Title may be repeated in multiple languages.

2— While the Identifier provides the definitive reference to the SRCM, it is typically unintelligible. The Title provides a convenient alternative readable form, but one which is not the definitive label. Examples: "English proficiency", "Schmiblick failure diagnostic level 4", "Demonstrates conflict resolution skills".

6.1.3 Description

Synopsis

```
description : bag of langStringType(4000),
   // SPM: 20 instance of langStringType in the bag
   // The parameter value is the SPM for the langString
```

Description

A human readable description of the competency map. This is an optional unstructured character string meant to be interpretable only by humans or unstructured full text indexing schemes.

Subclause 6.2.1 defines langStringType.

NOTES:

1—The Description may be repeated in multiple languages.

2—The description is typically more explicative than the title. Examples: "Proficiency in written and spoken English and use of English for meaningful oral or written expression.", "Performance of level 4 diagnostic as specified in IETM #SCMBLK007"

6.1.4 SRCM schema label

Synopsis

```
srcmSchemaLabel : characterstring(iso-10646-1),
      // SPM: 1000 characters
```

Description

This element contains a label for the schema that defines and controls this SRCM data.instance. If this element is omitted then a value of “IEEE 1484.20???" (tbd) shall be assumed. A conforming implementation may refuse to accept or process a SRCM instance if the value of this data element is specified, but is not the value specified in the Standard.

NOTES

1—Different label values may be used to signal application profiles with particular extensions or processing requirements.

2—This element does not describe the schema of the embedded metadata defined in 6.1.7. Every instance of embedded metadata, if any, should include its own schema description.

6.1.5 SRCM schema version

Synopsis

```
srcmSchemaVersion : characterstring(iso-10646-1),
      // SPM: 1000 characters
```

Description

Describes the version of the schema identified by `srcmSchemaLabel`. If this element is omitted then a value of “1.0” shall be assumed. A conforming implementation may refuse to accept or process a SRCM instance if the value of this data element is specified, but is not the value specified in the Standard.

NOTE—1—Different values may be used to signal application profiles with particular extensions or processing requirements.

6.1.6 Referential

Synopsis

```
referential : boolean, // default = false
```

Description

The `referential` data element indicates whether the map is self-contained or includes references to other maps. If the map contains references to other maps, this element shall be required and its value shall be true. This element shall be optional if the map does not contain references to other maps, in which case its default value shall be assumed to be false.

Notes

1— The entity that creates the map is responsible for setting the value of this element to accurately represent whether the map includes references to other maps.

2— This element is included to facilitate implementation efficiency in the processing of maps, e.g. to help determine whether a map has external dependencies without having to examine every node of the map.

6.1.7 Metadata

Synopsis

```
metadata :
   bag of anyType, // SPM: 10 of any type in the bag
```

Description

Optional embedded Metadata describing this SRCM.

If a metadata record is included, it is recommended that this record conform to IEEE 1484-12.1-2002: Standard for Learning Object Metadata (IEEE LOM). In such conforming records, the version of the Metadata Specification Standard is given in the meta-metadata element of the metadata record. Some of the data elements defined in IEEE LOM are not relevant for a SRCM and should be omitted. Profiles shall determine which metadata, if any, must be used for conformance with the profile.

NOTES:

1—Useful metadata defined in the IEEE LOM include additional identification as an entry in one or more catalogues, information about the author, publisher, the creation date, and the coverage (in the sense of the Dublin Core as adopted by the IEEE LOM.) The Relation element may be used to relate a definition to a prior version of the definition, and one or more Classification elements may be used to indicate where this particular map fits in a larger SRCM, map, model or ontology of competencies or educational objectives. Classification elements may be repeated to reference different models. For example, the SRCM might represent a map of a competency as defined in O*NET model, as well as a map of a similar competency defined in a company's custom competency model.

2—More than one metadata record is allowed in the bag, but if there is more than one record each record should conform to a different metadata specification. An implementation must accept any metadata record that it cannot interpret, but it is not required to interpret such metadata records.

3—A particular binding specification or application profile may impose additional restrictions or requirements.

6.1.8 Graph

Synopsis

```
graph : record :
(
   defaultEntryNode: localIdentifierType,
   entryNodes: bag of localIdentifierType,
   nodes : bag of nodeType,
      // SPM: (To be discussed. Specify in profiles?)
),
```

Description

A mandatory element. This element contains the actual map in the form of a collection of directed graph nodes. Any node without a parent is an entry node into the graph.

NOTES:

1— A binding might require a particular ordering of the elements. Such ordering does not imply any meaningful order.

2— Some competency models prescribe an ordering or precedence for competencies. This typically means that competencies must be mastered in a specific order. This, in turn, means that there are dependencies between the competencies so that mastery in A is a requirement for mastery in B, B is a requirement for mastery of C, and so on. This can be described in the DAG by making A a child of B, and B a child of C, etc. without adding an ordering property.

6.1.8.1 Default entry node

Synopsis

```
defaultEntryNode: localIdentifierType,
```

Description

This element specifies the default entry node into the graph. It is optional, but an application profile may require it to support applications that require a single entry node even if the graph contains more than one entry node.

6.1.8.2 Entry nodes

Synopsis

```
entryNodes: bag of localIdentifierType,
```

Description

This element contains a complete and exact list of the nodes that do not have at least one parent. There is no intrinsic order to the list.

Note – This element is provided to accelerate processing by not requiring a traversal of the entire node collection to identify the entry nodes.

6.1.8.3 Nodes

Synopsis

```
nodes : bag of nodeType,
```

Description

Unordered collection that contains all the nodes that make up the DAG. There is no intrinsic order to the nodes, other than the order defined by parent-child relationships. NodeType is defined in clause 6.2.1

6.1.9 Extensions

Synopsis

```
extensions : anyType,
```

Description

Application specific extensions to the data model. The extensions may not conflict with a binding of the data model. Extensions may not replace or duplicate elements defined in the data model.

NOTES:

1—The extensions should be defined by an application profile.

6.2 Data types

The following data types are used in conjunction with the data elements described in Clauses 6.1 and 6.2.

6.2.1 Node type

Synopsis

```
nodeType = record :
   (
      nodeId :
         localIdentifierType,
            // SPM: 1000 characters
      rcdReference :
         longIdentifierType,
            // SPM: 4000 characters;
      classLabel : record
```

```
            (
               model : longIdentifierType,
               token : characterstring,
               bag of langStringType(250),
               // SPM: 20 instances of langStringType in the bag
               // The parameter value is the SPM for the
               // langString
            )
         title :
            bag of langStringType(1000),
               // SPM: 20 instances of langStringType in the bag
               // The parameter value is the SPM for the
               // langString
         description :
            bag of langStringType(2000),
               // SPM: 20 instances of langStringType in the bag
               // The parameter value is the SPM for the
               // langString
         parents :
            bag of localIdentifierType,
               // nil if no parent
         children :
            bag of record
            (
               nodeRef: localIdentifierType,
               weight: real(10,7), // range (0..1) default = 1,
               dataRequired: boolean, // default = true;
               proficiencyRequired: proficiencyScoreType,
               proficiencyDesired: proficiencyScoreType,
                  // range (0..1) -- default = child's own value
            )
         symLink : longIdentifierType,
            // must be nil or omitted if children is not nil
         rules : record
            (
               proficiencyRequired: proficiencyScoreType,
               proficiencyDesired: proficiencyScoreType,
               rollupMethod : state
                  (all,any,fraction,units,mean,other),
               rollupParameter : choice(rollupMethod) of
                  (
                     all, any, mean : nil,
                     fraction: real(10,7), // range (0..1),
                     units: integer, // must be > 0
                     other: characterstring(iso-10646-1),
                        // SPM: 4000
                  )
            )
      ),
```

Description

This type defines the data model for a single node in the DAG that represents the map's topology. The following clauses describe the elements within each node record.

6.2.1.1 Node Identifier

Synopsis

```
nodeId : localIdentifier,
```

Description

The value of this mandatory element is a unique label that identifies the node. The Identifier is sufficient to reference the node in the graph. This Identifier shall be unique at least in the context of the SRCM. There are no semantics associated with the value nodeId element; it is just a key to reference the node.

Subclause 6.x.xxx defines `localIdentifierType`.

NOTES:

1—If there is any possibility that the map may be disaggregated or merged with other maps or fragments of maps, it is recommended that the Identifier be globally unique. The result of merging maps that contain nodes with conflicting identifiers is undefined.

6.2.1.2 RCD Reference

Synopsis

```
rcdReference :
   longIdentifierType,
      // SPM: 4000 characters
```

Description

This data element contains the identifier of the Reusable Competency Definition represented by this node. If the node exists only for the purpose of grouping other nodes in the map, this element may be omitted or its value may be nil. There are no semantics associated with the value of `rcdReference` element. It is just a key to reference some Reusable Competency Definition, wherever it may be.

6.2.1.3 Class Label

Synopsis

```
classLabel : record
      (
         model : longIdentifierType,
         bag of langStringType(250),
```

```
            // SPM: 20 instances of langStringType in the bag
            // The parameter value is the SPM for the
            // langString
         )
```

Description

This optional data element contains an optional reference to a model or vocabulary and a short text label for the class of node. This element is provided to allow the provision of a class label or similar label if the node is intended to represent a class in a particular competency model.

NOTES

1— An application may define that the model element is the identifier or locator for a vocabulary specification that complies with the IMS VDEX specification.

2— This element allows the representation of the mapping to various hierarchical models, such as Terminal learning objective (TLO) enabled by Enabling Learning Objectives (ELO)->ELO, Job competencies implying task competencies, behaviors vs. knowledge, and so on. The specification of node classes is outside the scope of this standard.

3— This element is multilingual to support the meaningful display of the node class in user interfaces. An application may define that only one language is meaningful for the label, in which case the label may be treated as a language-independent token.

6.2.1.4 Title

Synopsis

```
   title : bag of langStringType(1000),
      // SPM: 20 instance of langStringType in the bag
      // The parameter value is the SPM for the langString
```

Description

This optional data element contains a short human readable text label for the node.

If the node references a Reusable Competency Definition and this element has a value, implementations shall display the title as defined by this element instead of the title of the referenced Reusable Competency Definition. If the node references a Reusable Competency Definition and this element has no value, implementations should display the title as defined by the referenced Reusable Competency Definition. However, this is not a requirement because performance or other considerations may make this impractical. If no title is available for display, but it is necessary to display the node, the implementation shall define what to display.

6.2.1.5 Description

Synopsis

```
   description : bag of langStringType(2000),
      // SPM: 20 instance of langStringType in the bag
```

```
// The parameter value is the SPM for the langString
```

Description

This optional data element contains a human readable text description for the node.

If the node references a Reusable Competency Definition and this element has a value, implementations shall display the description as defined by this element instead of the description in the referenced Reusable Competency Definition. If the node references a Reusable Competency Definition and this element has no value, implementations should display the description as defined by the referenced Reusable Competency Definition, if available. However, this is not a requirement because performance or other considerations may make this impractical.

6.2.1.6 Parents

Synopsis

```
parents : bag of longIdentifierType,
```

Description

Identifiers of the parent nodes of this node in the directed graph topology. This element shall be null, empty or omitted if the node does not have any parent. If a node has no parent, it shall be included in the list of entry nodes (See Clause 6.1.8.2).

NOTES

1— In a DAG, a node may have more than one parent. A corollary is that several nodes may share one or more children.

2— A binding may require the addition of a placeholder element to contain each parent reference in the bag.

3— Although this is not recommended, a node that has one or more parent may be included in the list of entry nodes.

6.2.1.7 Children

Synopsis

```
bag of record
   (
      nodeRef: localIdentifierType,
      weight: real(10,7), // range (0..1) default = 1,
      dataRequired: boolean, // default = true;
      proficiencyRequired: proficiencyScoreType,
      proficiencyDesired: proficiencyScoreType,
         // range (0..1) -- default = child's own value
   )
```

Description

This data element represents the child nodes of this node in the directed graph topology. This element shall be null, empty or omitted if the node has no children. A node shall be prohibited to have children if a `symLink` to a different map is specified for the node.

6.2.1.8 Child Node Reference

Synopsis

```
children :
   bag of record
   (
      nodeRef: localIdentifierType,
      ...
   )
```

Description

This mandatory data element shall be a reference to another node in the same map instance. The reference may not be to an ancestor node of the node that owns the bag of children.

Note – If the bag of children is not empty, every child record in the bag must include a valid `nodeRef`. An ancestor node may not be referenced because that would result in an illegal loop in the DAG topology.

6.2.1.9 Child - Weight

Synopsis

```
children :
   bag of record
   (
      nodeRef: localIdentifierType,
      weight: real(10,7), // range (0..1) default = 1,
      ...
   )
```

Description

This data element is used only when the node is a child in a rollup operation where a weight assigned to the information associated with the child node is relevant. For example, if the rollup method is mean, this data element can be used to specify a relative weight among children. In the absence of a specific value, the default value of this element is assumed to be 1.

Because the value of the `weight` element may be different in the context of different parents, this element is defined in a rule associated with the parent.

Note – This element allows certain rollup operations where proficiency information is summarized in such a way that weights may influence the result. Weights are not relevant in some of the rollup methods defined in Clause 6.2.1.13.

6.2.1.10 Child – Data Required

Synopsis

```
children :
   bag of record
   (
      nodeRef: localIdentifierType,
      ...
      dataRequired: boolean, // default = true;
      ...
   )
```

Description

This data element is used only when the node is considered as a child in a rollup operation. If its value is false, then the child is included in rollups only if valid proficiency information is available for it. The child is ignored altogether if no valid proficiency information is available for it. In the absence of a specific value, the default value of this element is assumed to be true.

Because the value of the `required` element may be different in the context of different parents, this element is defined in a rule associated with the parent.

Note – This element allows certain rollup operations where proficiency information should be used if available, but ignored if not available. For example, an individual may be considered a good player if she is competent in all the sports for which proficiency data are available for her, even though the model may list additional sports in which she never participated.

6.2.1.11 Child - Proficiency Required

Synopsis

```
children :
   bag of record
   (
      nodeRef: localIdentifierType,
      ...
      proficiencyRequired: proficiencyScoreType,
      ...
   )
```

Description

This data item specifies a proficiency requirement for a child in the context of this parent node. It specifies that proficiency status can be assumed to be true if the available proficiency measure associated with the child node is at least a certain value. For valid comparisons, the

proficiency measure must be expressed in a compatible range. The range for `proficiencyRequired` is -1 to 1 inclusive. The default default value if not explicitly specified is 1.

The `proficiencyRequired` element defined in the child node's own rules shall be used if no overriding value is specified in this data element.

NOTES:

1— Because the value of the `proficiencyRequired` element may be different in the context of different parents, this data element allows `proficiencyRequired` to be specified in this data element associated with the parent.

2— Proficiency measure is also called "proficiency level", "score" or "success measure" in various specifications and Standards.

3— Application example involving rollup: Node RX references RCD X and specifies a proficiency required of 70%. However, node RX is a child of node RA that specifies that for child RX the proficiency required is 80%. When rolling up competency status information from RX into the competency status information for RA, the proficiency required used to evaluate whether a measure satisfies the requirement will be 80%. This is because the requirements specified for the context override the requirements specified for individual components of the context. On the other hand, if no rule associated with RA specifies a particular proficiency required for child RX, then the proficiency required defined in the node RX is used by default.

6.2.1.12 Child - Proficiency Desired

Synopsis

```
children :
   bag of record
   (
      nodeRef: localIdentifierType,
      ...
      proficiencyDesired: proficiencyScoreType
   )
```

Description

This data item specifies a proficiency requirement for a child in the context of this parent node. It specifies that proficiency status can be assumed to be true if the available proficiency measure associated with the child node is at least a certain value. For valid comparisons, the proficiency measure must be expressed in a compatible range. The range for `proficiencyDesired` is -1 to 1 inclusive. The default default value if not explicitly specified is 1.

The `proficiencyDesired` element defined in the child node's own rules shall be used if no overriding value is specified in this data element.

NOTE— Application example: The current node references RCD X and a competency evidence record for an individual also points to RCD X, and specifies that the individual's proficiency level

is 67%. If a `proficiencyRequired` of 60% (=0.6) is specified by for this node, the individual is not considered proficient according to this competency map. However, if a proficiency `proficiencyDesired` of 75% is specified, then a learning application might try to engage the individual in a learning activity to achieve the desired level of proficiency, even though the baseline required proficiency is already achieved.

6.2.1.13 Sym Link

Synopsis

```
symLink : longIdentifierType,
```

Description

Identifier referencing a map used to extend the current map by merging an entry node in that map with the current node. The binding of the identifier may include the specification of a particular entry node in the target map. If the target map has multiple entry nodes and no entry node is specified in this element, the default entry node in the target map shall be used as entry node. The merging rules are specified in clause x.x (TBD).

NOTES:

1— See conceptual overview for a detailed informative explanation of the map extension mechanism.

2— By using sym links, it is possible to keep SRCMs to an easily manageable size, and to reuse SRCMs in the context of other maps. For example, the same skill definition published by a state standards board may be included in various institution-specific competency models through this mechanism.

SymLink merging rules

When a sym link is used, the target node that is the entry point in the target map shall be treated as if it was merged with the referencing node when processing data from the tree. If any two data elements or attributes conflict between the nodes during processing, the element or attribute value of the referencing node shall override the value of the target node. Children of the target node, if any, shall be treated as children of the referencing node when traversing the graph and performing rollups. Deleting a referencing node shall have no effect on the target node. Changing values or data elements of the referencing node shall have no effect on the values or data elements of the target node. Changing values or data elements of the target node shall have no effect on the values or data elements of the referencing node. The only actual merging or override of values or data elements shall occur in the data space of the processing entity.

6.2.1.14 Rules

Synopsis

```
rules : record
```

```
(
   proficiencyRequired: proficiencyScoreType,
   proficiencyDesired: proficiencyScoreType,
   rollupMethod :  state(all,any,fraction,units,mean,other),
   rollupParameter : choice(rollupMethod) of
      (
         all, any, mean : nil,
         fraction: real(10,7), // range (0..1),
         units: integer, // must be > 0
         other: characterstring(iso-10646-1),
            // SPM: 4000
      )
)
```

Description

Rules that specify the data processing behavior of systems that use the SRCM.

If the rules element is not present in an instance, the default value for each the rule elements defined here shall be applied in processing rules. If the rules element is present but any of the rule elements is not specified, the default value for that element shall be assumed in processing rules.

6.2.1.15 Proficiency required

Synopsis

```
proficiencyRequired: proficiencyScoreType,
```

Description

This data item specifies a proficiency requirement. It specifies that proficiency status can be assumed to be true if the available proficiency measure for the referenced competency definition is at least a certain value. For valid comparisons, the proficiency measure must be expressed in a compatible range. The range for `proficiencyRequired` is -1 to 1 inclusive. The default default value if not explicitly specified is 1.

The `proficiencyRequired` element defined in the node's rules shall be used if the node is considered independently of any parent or if no overriding value is specified in the rules of the parent node being considered in an operation. Because the value of the `proficiencyRequired` element may be different in the context of different parents, a value for `proficiencyRequired` may also be specified in the child specification rule associated with each parent of this node. For rollup processes, if a value for `proficiencyRequired` is specified in the child specification of the parent node, that value shall be used instead of the value of the proficiencyRequired element of the child node.

NOTES:

1— Proficiency measure is also called "proficiency level", "score" or "success measure" in various specifications and Standards. Some semantic mapping may be required for integration with those specifications and standard.

2— Application example: The current node references RCD X and a competency evidence record for an individual also points to RCD X, and specifies that the individual's proficiency level is 67%. If a proficiency required of 70% (=0.7) is specified by for this node, the individual is not considered proficient according to this competency map.

3— Application example involving rollup: Node RX references RCD X and specifies a proficiency required of 70%. However, node RX is a child of node RA that specifies that for child RX the proficiency required is 80%. When rolling up competency status information from RX into the competency status information for RA, the proficiency required used to evaluate whether a measure satisfies the requirement will be 80%. This is because the requirements specified for the context override the requirements specified for individual components of the context. On the other hand, if no rule associated with RA specifies a particular proficiency required for child RX, then the required defined for RX is used by default.

6.2.1.16 Proficiency desired

Synopsis

```
proficiencyDesired: proficiencyScoreType,
```

Description

This data item specifies a proficiency requirement. It specifies that proficiency status can be assumed to be true if the available proficiency measure for the referenced competency definition is at least a certain value. For valid comparisons, the proficiency measure must be expressed in a compatible range. The range for `proficiencyDesired` is -1 to 1 inclusive. The default default value if not explicitly specified is 1.

The `proficiencyDesired` element defined in the node's rules shall be used if the node is considered independently of any parent or if no overriding value is specified in the rules of the parent node being considered in an operation. Because the value of the `proficiencyDesired` element may be different in the context of different parents, a value for `proficiencyDesired` may also be specified in the child specification rule associated with each parent of this node. For rollup processes, if a value for `proficiencyDesired` is specified in the child specification of the parent node, that value shall be used instead of the value of the proficiencyDesired element of the child node.

NOTE— Application example: The current node references RCD X and a competency evidence record for an individual also points to RCD X, and specifies that the individual's proficiency level is 67%. If a proficiency required of 60% (=0.6) is specified by for this node, the individual is not considered proficient according to this competency map. However, if a proficiency desired of 75% is specified, then a learning application might try to engage the individual in a learning activity to achieve the desired level of proficiency, even though the baseline required proficiency is already achieved.

6.2.1.17 Rollup method

Synopsis

```
rollupMethod : state(all,any,fraction,units,mean,other),
```

Description

This data item applies only when the node is a parent. It specifies how to roll up proficiency information specified by children of the node. The defined methods are:

- all -- Proficiency must be achieved for all the competencies represented by children nodes in order to consider that proficiency is achieved for the competency represented by this node. This is the assumed default value if a rollup method is not specified.

- any -- Proficiency must be achieved for any of the competencies represented by children nodes in order to consider that proficiency is achieved for the competency represented by this node.

- fraction -- Proficiency must be achieved for at least specified fraction of the competencies represented by children nodes in order to consider that proficiency is achieved for the competency represented by this node. The fraction value is expressed as a floating point number in the range 0 to 1 inclusive. 0 means that no child proficiency is required. 1 is the equivalent of specifying that the rollup method is all. 0.5 means that proficiency must be achieved for at least 50% of the child nodes.

- units -- Proficiency must be achieved for at least the specified number of the competencies represented by children nodes in order to consider that proficiency is achieved for the competency represented by this node. The number is a positive integer. 0 means that no child proficiency is required. 1 is the equivalent of specifying that the rollup method is any.

- mean -- The proficiency measure for the parent node is determined by averaging the proficiency levels for the competencies represented by children nodes. The proficiency status for the parent node can then be determined by comparing that level with the proficiencyRequired value for the parent node. If no proficiency level is available for a child node, the proficiency status (boolean) is used instead, with proficient = 1 and not proficient or unknown = 0.

- other -- Another proficiency method is to be used. The other method must be specified by an application profile and is not defined by the Standard.

By default, the rollup method is "all". No available competency data or a value of "unknown" is considered the same as "not proficient" for the purpose of rollup. If proficiency status but no proficiency measure is available for any node, the proficiency measure value is assumed to be 1 if proficient, and 0 if not proficient.

NOTES:

1— Proficiency measure is also called "proficiency level", "score" or "success measure" in various specifications and Standards.

6.2.1.18 Rollup parameter

Synopsis

```
rollupParameter : choice(rollupMethod) of
   (
      all, any, mean : nil,
      fraction: real(10,7), // range (0..1),
      units: integer, //
      other: characterstring(iso-10646-1),
         // SPM: 4000
   )
```

Description

Parameter used only if the specified rollup method is fraction, units or other. The type of parameter depends on the rollup method.

If the rollup method is "other", the parameter is a container for data of any type, and an application profile must specify the method and the data.

6.2.2 Any type

Synopsis

```
type anyType = (unspecified);
```

Description

This data type represents any type not specified in this Standard. This Standard does not require an implementation to process data elements of this type when encountered in a RCD instance. The data types implemented in a SRCM data element with type anyType shall not preclude interoperable bindings of the entire SRCM instance.

NOTES

1— If implementations specify or require data elements for which the type is defined in this Standard as anyType, this Standard recommends that the implementations provide the means to interpret and validate the implementation-specific data. For example, an implementation that uses an XML binding should include a valid XML schema that can be referenced in RCD instances bound in XML documents, and the schema should be documented.

2—The data types used in the container should be defined by an application profile.

Examples

An application profile might specify an additional data element for "desired proficiency" to use along with "required proficiency" in SRCM instances that represent a position competency model for hiring.

An application profile might specify a rollup method that is not defined in the standard, and embed the specification for that rollup method as an extension in a standard-conformant instance.

An application profile might add references to contextual information that can be used to make an instance of the standard data model more context-specific.

An application profile might add metadata to identify the identity that set a proficiency level requirement, and to reference the enterprise policy that is the source of the requirement.

6.2.3 LangString type

Synopsis

```
type langStringType(length) =
   record
   (
      language :
         languageType,
      string :
         characterstring(iso-10646-1)
            // SPM: The length parameter
   );
```

Description

This data type consists of a language specification for a string and the string itself.

Examples

The following are three examples of localized strings: `"Information Technology"` in French, `"localization"` in British English, and `"xxx"` in Japanese hiragana.

```
( "fr", "Technologies de l'information" ),
( "en-GB", "localisation" ),
( "jp-JP-jisx208", "xxx" ),
```

6.2.3.1 Language

Synopsis

```
language :
   characterstring(iso-646),
      // SPM: 250 characters
```

Description

The `language` data element specifies the language of the localized string. The format of this data type is a character string consisting of a required language code followed by multiple, optional, hyphen-prefixed subcodes (see examples below).

The following rules apply to the language code part of the character string:

– 2-letter codes are defined by ISO 639–1.
– 3-letter codes are defined by ISO 639–2.
– The value prefix `"i"` is reserved for registrations defined by the Internet Assigned Numbers Authority (IANA).
– The value prefix `"x"` is reserved for private use.

The following rules apply to the first subcode part of the character string:

– 2-letter subcodes are ISO 3166–1 alpha-2 country codes.
– Subcodes of from 3 to 8 letters are registered with IANA.

Rules for additional subcodes are unspecified, except that the length of any subcode cannot exceed 8 characters.

NOTE—The language code is often given in lower case and the subcodes (if any) in upper case. However, the values are case insensitive.

Examples

```
"en-GB"
"de"
"fr-CA"
"it"
"grc" (Ancient Greek, until 1453)
"en-US-NY-NY"
"eng-GB-cockney"
"map-PG-buin" (Austronesian - Papua New Guinea Buin)
"gem-US-PA"
```

6.2.3.2 String

Synopsis

```
string :
   characterstring(iso-10646-1),
      // SPM: The length parameter
```

Description

The `string` data element contains the text of the localized string.

6.2.4 Local identifier type

Synopsis

```
type localIdentifierType = characterstring(iso-10646-1)
        // SPM: 4000 characters
```

Description

This data type is an identifier (a label) associated with an object that is intended to be unique within the context of usage of the object. The character string shall conform to the syntax for Uniform Resource Identifiers (URIs) as defined by RFC 2396.

NOTES

1—This Standard recommends that the URI be a globally unique identifier in the form of a Uniform Resource Name (URN) (see RFC 2141 [A5]) or a Handle as defined in the Handle System [Ax].

2— Depending on the binding, these data elements may be implemented as explicit or implicit. The binding may be a combination of explicit and implicit representations of these data elements. For example, in an XML document they may be implicit in the structure of the document if the shape of the graph is a simple tree.

3— Depending on the implementation, the values in the parents and children collections may be implicit, may be explicit and literal or may be an implementation-specific identifier or pointer. However, any conforming implementation of a Reusable Competency Map must represent no less than the information specified in this Standard for any component that is present in the implementation. For example, even it is a node is represented by a relative pointer to achieve data compression in a particular binding, the identifier for that node must be resolvable to a conformant identifier type when the tree instance is transmitted to another conformant implementation.

6.2.5 Long identifier type

Synopsis

```
type longIdentifierType =
   record (
      catalog: characterstring(iso-10646-1),
         // SPM: 4000 characters
      entry: characterstring(iso-10646-1)
         // SPM: 4000 characters
   );
```

Description

This data type is an identifier (a label) associated with an object that is intended to be unique within the context of usage of the object. The character string shall conform to the syntax for Uniform Resource Identifiers (URIs) as defined by RFC 2396.

NOTE—This Standard recommends that the URI be a globally unique identifier in the form of a Uniform Resource Name (URN) (see RFC 2141 [A5]) or a Handle as defined in the Handle System [Ax].

6.2.6 ProficiencyScoreType

Synopsis

```
type proficiencyScoreType = real(10,7);
   // range -1..1
```

Description

This data type is a proficiency score on a continuous scale constrained to the values -1 to 1 inclusive. Negative scores are allowed to permit the representation of "negative proficiency", e.g. if a person is actually dangerously inept in the execution of a task.

NOTES

1— Negative scores are allowed to permit the representation of "negative proficiency", e.g. if a person is actually dangerously inept in the execution of a task.

2— This type maps exactly to the scaled score type defined in IEEE 1484.11.1.

3— Rollups and comparisons of scores from different sources require a common scale. Various proficiency scales or grading methods may be mapped to and from the ProficiencyScoreType. For discontinuous scales, the mapping typically involves ranges or thresholds. Such mappings are outside the scope of this standards.

Annex B

[A2] IEEE 1484.11.2–2003, Standard for Learning Technology—ECMAScript Application Programming Interface for Content to Runtime Services Communication.

[A3] IEEE 100, The Authoritative Dictionary of IEEE Standards Terms, Seventh Edition.

[A5] RFC 2141, "URN Syntax," Network Working Group, May 1997.

IMSRDCEO IMS Global Learning Consortium, Inc. (2002). IMS Reusable Definition of Competency or Educational Objective, Version 1.0

O*NET http://online.onetcenter.org/ (or http://www.access.gpo.gov/o_net/datadict/datadict.pdf)

Ostyn http://ltsc.ieee.org/doc/wg20/CompDefInit.doc (Base document from P1484.20)

PASS http://www.ous.edu/pass/Standards/admission.html (Oregon Proficiency-based Admissions Standards System)

SCANS http://www.tier.net/tcenters/scans.htm (Secretary's Commission on Achieving Necessary Skills: Competencies)

SCORM http://www.adlnet.org (ADL SCORM)

TATS http://www.adtdl.army.mil/atdls.htm (Total Army Training System)

Annex B

(informative)

Sample XML Binding

This sample shows an XML instance and the corresponding schema for an XML binding. This is a sample only. The instance shows that RCDs are known to exist for 3 competencies defined by different agencies: "Can drive a car safely", "Can steer a car and control its speed" and "Knows and understands road signs and markings", and that the first competency is composed of the other two. No rules are specified because the default rules are good enough.

```
<?xml version="1.0" encoding="UTF-8"?>
<simpleCompetencyMap xmlns="proposedForIEEE-LTSC-WG20/simpleReusableCompetencyMap"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="proposedForIEEE-LTSC-WG20/simpleReusableCompetencyMap simple-
competency-map-draft.xsd">
  <mapId>123356/123.45tyu/345xyz</mapId>
  <schemaLabel>SRCM0.1.0</schemaLabel>
  <schemaVersion>1.0</schemaVersion>
  <graph>
    <defaultEntryNode nodeRef="someIdOrOther6"/>
    <entryNodes>
      <entryNode nodeRef="someIdOrOther6"/>
    </entryNodes>
    <node nodeId="someIdOrOther6">
      <rcdRef ref="http://somelicensingdept.gov/ref/driver/minimal"/>
      <title>
        <langString lang="en">Can drive a car safely</langString>
        <langString lang="fr">Capable de conduire une voiture prudemment</langString>
      </title>
      <children>
        <child nodeRef="someIdOrOther7"/>
        <child nodeRef="someIdOrOther8"/>
      </children>
    </node>
    <node nodeId="someIdOrOther7">
      <rcdRef ref="http://somelicensingdept.gov/ref/driver/sk1"/>
      <title>
        <langString lang="en">Can steer a car and control its speed</langString>
        <langString lang="fr">Peut diriger et controler la vitesse</langString>
      </title>
      <parents>
        <parent nodeRef="someIdOrOther6"/>
      </parents>
    </node>
    <node nodeId="someIdOrOther8">
      <rcdRef ref="http://usdot.gov/rcdsroadsigns2005"/>
      <title>
        <langString lang="en">Knows and understands road signs and markings</langString>
        <langString lang="fr">Connaît et comprend les signaux routiers</langString>
      </title>
      <parents>
        <parent nodeRef="someIdOrOther6"/>
      </parents>
    </node>
  </graph>
</simpleCompetencyMap>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:srcm="proposedForIEEE-LTSC-
WG20/simpleReusableCompetencyMap" targetNamespace="proposedForIEEE-LTSC-
WG20/simpleReusableCompetencyMap" elementFormDefault="qualified"
attributeFormDefault="unqualified" version="experimental0.4">
  <xs:annotation>
    <xs:documentation>
        Copyright 2005, 2006 Claude Ostyn - All rights reserved.
         Schema for a simple reusable competency map implementing a proposed standard.
```

```
          See proposed standard text at http://ostyn.com/resources.htm for details.
          A competency map may reference a Reusable Competency Definition
          (RCD) or be standalone.
          The map represents a hierarchy of Competency Definitions.  It does not include
          the RCDs. When used  as a map of RCDs, different map instances
          may represent different beliefs as to how those RCDs are related.
          The nodes in the map are organized as a  Directed Acyclic Graph (DAG),
          but the graph can be easily used to represent a tree with a single root node.
          The graph is represented as a flat list of nodes,
          each of which has a parent and/or and children list (which may be empty).
          The human-readable strings in the map may be specified in multiple
          languages. This introduces a very small amount of overhead if you
          do not need that feature, e.g. instead of something like
              (title)my title(/tilte)
          you use
             (title)
               (langString)my title(/langString)
             (/title)
          This is a strawman schema and should not be used for conformance.
        </xs:documentation>
    </xs:annotation>
    <!--  -->
    <!-- Reusable elements -->
    <!-- -->
    <xs:element name="simpleCompetencyMap" type="srcm:competencyDAGraphType">
      <xs:annotation>
        <xs:documentation>
          Complete map, consisting of front matter and  the actual graph.
        </xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="classLabel" type="srcm:classLabelType">
      <xs:unique name="UniqueClassLabelLanguageInstance">
        <xs:selector xpath="./srcm:langString"/>
        <xs:field xpath="@lang"/>
      </xs:unique>
    </xs:element>
    <xs:element name="description" type="srcm:langStringBagType">
      <xs:unique name="UniqueDescriptionLanguageInstance">
        <xs:selector xpath="./srcm:langString"/>
        <xs:field xpath="@lang"/>
      </xs:unique>
    </xs:element>
    <xs:element name="rcdRef" type="srcm:rcdRefType"/>
    <xs:element name="title" type="srcm:langStringBagType">
      <xs:unique name="UniqueTitleLanguageInstance">
        <xs:selector xpath="./srcm:langString"/>
        <xs:field xpath="@lang"/>
      </xs:unique>
    </xs:element>
    <!--  -->
    <!-- Main  type definitions -->
    <!-- -->
    <!-- The whole shebang -->
    <xs:complexType name="competencyDAGraphType">
      <xs:sequence>
        <xs:element name="mapId" type="xs:anyURI" nillable="false"/>
        <xs:element ref="srcm:rcdRef" minOccurs="0"/>
        <xs:element ref="srcm:classLabel" minOccurs="0"/>
        <xs:element ref="srcm:title" minOccurs="0"/>
        <xs:element ref="srcm:description" minOccurs="0"/>
        <xs:element name="schemaLabel" type="xs:string" default="SRCM0.1.0"/>
        <xs:element name="schemaVersion" type="xs:string" default="1.0"/>
        <xs:element name="referential" type="xs:boolean" default="false" minOccurs="0"/>
        <xs:element name="metadata" type="srcm:metadataType" minOccurs="0"/>
        <xs:element name="graph" type="srcm:graphType">
          <xs:key name="KeyNodeIdentifier">
            <xs:selector xpath=".//srcm:node"/>
            <xs:field xpath="@nodeId"/>
          </xs:key>
        </xs:element>
```

```
      <xs:group ref="srcm:groupAny"/>
    </xs:sequence>
    <xs:attributeGroup ref="srcm:attGroupAny"/>
  </xs:complexType>
  <xs:complexType name="graphType">
    <xs:annotation>
      <xs:documentation>
        The DAG shaped graph, as a flat list of nodes.
        No ordering is implied, other than the directionality
        of the graph determined by parent to child associations..
        </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="defaultEntryNode">
        <xs:complexType>
          <xs:attribute name="nodeRef" type="srcm:localIdentifierType" use="required"/>
        </xs:complexType>
      </xs:element>
      <xs:element name="entryNodes">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="entryNode" maxOccurs="unbounded">
              <xs:complexType>
                <xs:attribute name="nodeRef" type="srcm:localIdentifierType"
                      use="required"/>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="node" type="srcm:nodeType" maxOccurs="unbounded"/>
      <xs:group ref="srcm:groupAny"/>
    </xs:sequence>
    <xs:attributeGroup ref="srcm:attGroupAny"/>
  </xs:complexType>
  <xs:complexType name="nodeType">
    <xs:annotation>
      <xs:documentation>
        Any node in the graph.
        </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element ref="srcm:classLabel" minOccurs="0"/>
      <xs:element ref="srcm:rcdRef" minOccurs="0"/>
      <xs:element ref="srcm:title" minOccurs="0"/>
      <xs:element ref="srcm:description" minOccurs="0"/>
      <xs:element name="metadata" type="srcm:metadataType" minOccurs="0"/>
      <xs:element name="parents" type="srcm:parentsType" minOccurs="0"/>
      <xs:element name="children" type="srcm:childrenType" minOccurs="0"/>
      <xs:element name="symLink" type="xs:anyURI" minOccurs="0"/>
      <xs:element name="rules" type="srcm:rulesType" minOccurs="0"/>
      <xs:group ref="srcm:groupAny"/>
    </xs:sequence>
    <xs:attribute name="nodeId" type="srcm:localIdentifierType" use="required"/>
    <xs:attributeGroup ref="srcm:attGroupAny"/>
  </xs:complexType>
  <!-- -->
  <!-- Various utility types used in the main types -->
  <!-- -->
  <xs:complexType name="classLabelType">
    <xs:annotation>
      <xs:documentation>
        Multilingual label with optional attributes to specify a model
        and token. The model can identify a modeling method or a vocabulary.
        The token can contain a language-independent name for the class.
        </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="srcm:langStringBagType">
        <xs:attribute name="model" type="xs:anyURI" use="optional"/>
        <xs:attribute name="token" type="xs:token" use="optional"/>
```

```
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="rcdRefType">
    <xs:annotation>
      <xs:documentation>
        Reference to a rcd (typically a URN or handle) .
        </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:group ref="srcm:groupAny"/>
    </xs:sequence>
    <xs:attribute name="ref" type="xs:anyURI"/>
    <xs:attributeGroup ref="srcm:attGroupAny"/>
  </xs:complexType>
  <xs:complexType name="parentsType">
    <xs:annotation>
      <xs:documentation>
        Collection of associations with parent nodes. No order is implied.
        These are empty elements but an application may use
        the extension facility to include additional elements inside.
        </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="parent" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:attribute name="nodeRef" type="srcm:localIdentifierType" use="required"/>
        </xs:complexType>
      </xs:element>
      <xs:group ref="srcm:groupAny"/>
    </xs:sequence>
    <xs:attributeGroup ref="srcm:attGroupAny"/>
  </xs:complexType>
  <xs:complexType name="childrenType">
    <xs:annotation>
      <xs:documentation>
        Collection of associations with child nodes. Application profiles
        determine whether those should be considered ordered or not.
        The elements include association qualifiers.
        The extension facility may be used to add application-specific
        qualifiers for the association.
        </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="child" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="weight" type="srcm:weightType" default="1.0"
minOccurs="0"/>
            <xs:element name="dataRequired" type="xs:boolean" default="true"
minOccurs="0"/>
            <xs:element name="proficiencyRequired" type="srcm:proficiencyScoreType"
minOccurs="0"/>
            <xs:element name="proficiencyDesired" type="srcm:proficiencyScoreType"
minOccurs="0"/>
            <xs:group ref="srcm:groupAny"/>
          </xs:sequence>
          <xs:attribute name="nodeRef" type="srcm:localIdentifierType" use="required"/>
          <xs:attributeGroup ref="srcm:attGroupAny"/>
        </xs:complexType>
      </xs:element>
      <xs:group ref="srcm:groupAny"/>
    </xs:sequence>
    <xs:attributeGroup ref="srcm:attGroupAny"/>
  </xs:complexType>
  <xs:complexType name="rulesType">
    <xs:annotation>
      <xs:documentation>
        Rules associated directly with a node.
        See document.
        </xs:documentation>
```

```
    </xs:annotation>
    <xs:sequence>
      <xs:element name="proficiencyRequired" type="srcm:proficiencyScoreType"
minOccurs="0"/>
      <xs:element name="proficiencyDesired" type="srcm:proficiencyScoreType"
minOccurs="0"/>
      <xs:element name="rollupMethod" type="srcm:rollupMethodType" default="all"
minOccurs="0"/>
      <xs:element name="rollupParam" type="xs:string" minOccurs="0"/>
      <xs:group ref="srcm:groupAny"/>
    </xs:sequence>
    <xs:attributeGroup ref="srcm:attGroupAny"/>
  </xs:complexType>
  <!--  -->
  <!-- Generic type definitions usable in various schemas -->
  <!-- -->
  <xs:simpleType name="localIdentifierType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        URI rather than id to allow custom identifier syntax not
        restricted to NCName syntax.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:anyURI"/>
  </xs:simpleType>
  <xs:complexType name="metadataType" abstract="false">
    <xs:annotation>
      <xs:documentation xml:lang="en">
         Placeholder for zero or more metadata instances of any type.
         Also provides a basic name-value pair collection for named
         langString values. If the language feature is not required,
         the language attribute may be omitted as in
         (meta tag="my name")
            (langString)the string(/langString)
         (/meta)
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="meta" minOccurs="0" maxOccurs="unbounded">
        <xs:complexType>
          <xs:complexContent>
            <xs:extension base="srcm:langStringBagType">
              <xs:attribute name="tag" type="xs:string" use="required"/>
            </xs:extension>
          </xs:complexContent>
        </xs:complexType>
        <xs:unique name="UniqueMetaLanguageInstance">
          <xs:selector xpath="./langString"/>
          <xs:field xpath="@lang"/>
        </xs:unique>
      </xs:element>
      <xs:group ref="srcm:groupAny"/>
    </xs:sequence>
    <xs:attributeGroup ref="srcm:attGroupAny"/>
  </xs:complexType>
  <xs:simpleType name="real7Type">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        As explained in IEEE 1484.11.1-2004, Annex B.1 Real data type
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:decimal"/>
  </xs:simpleType>
  <xs:complexType name="proficiencyScoreType">
    <xs:annotation>
      <xs:documentation xml:lang="en">
        Can be used in 3 ways:
        a. As empty element with the scaled attribute set to a scaled score
          conformant to IEEE 1484.11.1
        b. Without the scaled attribute but with additional application-
           specific elements specifying an application specific
```

```
            scale and value data.
          c. Combination of scaled value attribute and application-
              specific scale and value data.
        </xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:group ref="srcm:groupAny"/>
      </xs:sequence>
      <xs:attribute name="scaled" type="srcm:scaledScoreType" use="optional"/>
      <xs:attributeGroup ref="srcm:attGroupAny"/>
    </xs:complexType>
    <xs:simpleType name="scaledScoreType">
      <xs:annotation>
        <xs:documentation xml:lang="en">
          As defined in IEEE 1484.11.1-2004
        </xs:documentation>
      </xs:annotation>
      <xs:restriction base="srcm:real7Type">
        <xs:minInclusive value="-1"/>
        <xs:maxInclusive value="1"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="weightType">
      <xs:annotation>
        <xs:documentation xml:lang="en">
          Weight as a fuzzy logic multiplier (0..1)
        </xs:documentation>
      </xs:annotation>
      <xs:restriction base="srcm:real7Type">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="1"/>
      </xs:restriction>
    </xs:simpleType>
    <!-- -->
    <!-- Internationalization support -->
    <!-- -->
    <xs:complexType name="langStringBagType">
      <xs:annotation>
        <xs:documentation xml:lang="en">
          Unordered collection of 0 or more strings that should carry
          the same meaning in different languages. Each string
          must be  tagged with a different lang  attribute.
          One string may omit the lang attribute.
        </xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:element name="langString" type="srcm:langStringType"
         minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
    <xs:complexType name="langStringType">
      <xs:annotation>
        <xs:documentation xml:lang="en">
         String element with an optional language attribute.
         Application-specific additional attributes are allowed.
         See proposed standard document text.
         </xs:documentation>
      </xs:annotation>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="lang" type="xs:string" default=""/>
          <xs:attributeGroup ref="srcm:attGroupAny"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
    <!--  -->
    <!-- Simple types  -->
    <!--  -->
    <!--  -->
    <!-- Attribute type definitions  -->
    <!--  -->
```

```
  <xs:simpleType name="rollupMethodType">
    <xs:annotation>
      <xs:documentation>
        Rollup method applied by a parent node when
        rolling up data from its children. See text.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:token">
      <xs:enumeration value="all"/>
      <xs:enumeration value="any"/>
      <xs:enumeration value="fraction"/>
      <xs:enumeration value="units"/>
      <xs:enumeration value="mean"/>
      <xs:enumeration value="other"/>
    </xs:restriction>
  </xs:simpleType>
  <!--  -->
  <!-- Extensibility support  -->
  <!--  -->
  <xs:group name="groupAny">
    <xs:annotation>
      <xs:documentation>
         Any namespaced element from any namespace may be included
         where this placeholder appears.  The namespace for the imported element
         must be defined in the instance, and the schema should be imported.
      </xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:any namespace="##other" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:group>
  <xs:attributeGroup name="attGroupAny">
    <xs:annotation>
      <xs:documentation>
         Any namespaced attribute from any namespace may be included
         where this placeholder appears.  The namespace for the imported element
         must be defined in the instance, and the schema should be imported.
      </xs:documentation>
    </xs:annotation>
    <xs:anyAttribute namespace="##other" processContents="lax"/>
  </xs:attributeGroup>
</xs:schema>
```