SCOR DIGITAL STANDARD

The ASCM SCOR Digital Standard Information Model

ASSOCIATION FOR SUPPLY CHAIN MANAGEMENT

Introduction to the SCOR Digital Standard Information Model (SDSIM)

Since SCOR was introduced by the Supply Chain Council in 1996, it has become the global standard model for supply chain management. The combination of performance, process, practices and people have provided businesses an unparalleled reference as to how to position their supply chains for competitive advantage at multiple layers. Now with the proliferation of new technology and the explosion of data, new information sources are being integrated, linked to established content, and analyzed by digital products and services that offer advanced capabilities. These digital tools can predict, prescribe and assist in new forms of decision-making. This, in turn, enables companies to sense emerging risks and opportunities across their extended supply chains.

As organizations are reviewing and revising their supply chain processes and capabilities to support new digital business models, there is a recognized need to enrich the SCOR model with a *linked data* information model to support digital strategies. Often, multiple digitization initiatives are executed in parallel, mirrored by similar efforts by partners and competitors. A unified and consistent approach, the minimization of repetitious work at a conceptual level, and the integration of different data sources are important considerations. Supporting professionals (both supply chain and information technology) in clarifying the intricacies of domain or organizational data landscapes as well as providing a unified understanding and format of processes, metrics and other entities are prerequisites for collaborating and exchanging data.

Supply chain and information technology professionals who achieve a high level of performance invariably must become expert users of supply chain data. Further, they must be able to convert *linked data* into information and understand how various data elements and information classes work with or against each other. In doing so, they use purpose-built information models, often without realizing it. A simple example is to combine data from should-be calculations with current values to identify which items have the largest quantity or value discrepancies. These information models are at the center of information technology. In fact, they are a crucial part of every enterprise resources planning (ERP) system.

This section of the SCOR Digital Standard will introduce a unified information model architecture, capturing all key elements of the supply and value chain domain in a formal and machine-readable format. Stating the combined domain knowledge, compiled and maintained by ASCM and its members, in an unambiguous and digital manner provides a formal foundation for most problems of the domain’s digital transition. Data-centric approaches are common solutions to emerging challenges, such as advanced automation, machine learning or end-to-end collaboration. To gather, unify and communicate (link) data from different sources, departments and organizations without loss of content, meaning or functionality requires a universal and flexible language that is understood by all participants, humans and machines alike. A well-defined information model can serve as such a language.

Linked Data Conceptual Overview

In 2001, Tim Berners-Lee, James Hendler and Ora Lassila laid out their expectation of how the World Wide Web will eventually extend to become a Semantic Web[1]. The simple extension of web resources with structured, well-defined data will give meaning to resources that were previously only decipherable by humans. In their view, identifying these data resources uniquely with uniform resource identifiers (URIs) and providing links to other resources would be the first steps toward creating a web of data that could be processed directly by machines. Ultimately, the Semantic Web would better enable computers and people to work together.

The methods and technologies used to publish data artifacts according to these principles are known as Linked Data. The name highlights the data's propensity to interconnect effortlessly with other data objects by establishing links among them. Fundamentally, this technology is based on simple statements, representing declarative sentences made of a subject, predicate and object, which together are called triples.

```
Make-to-Stock (subject) => has observable metric (predicate) => Current Make Volume (object)
```

This example points out that the SCOR Process Make-to-Stock (sM1), which is the subject, acknowledges Current Make Volume (AG.3.38), the object, as an applicable metric. This is expressed by the predicate, has observable metric. Combining multiple statements like this creates a graph.

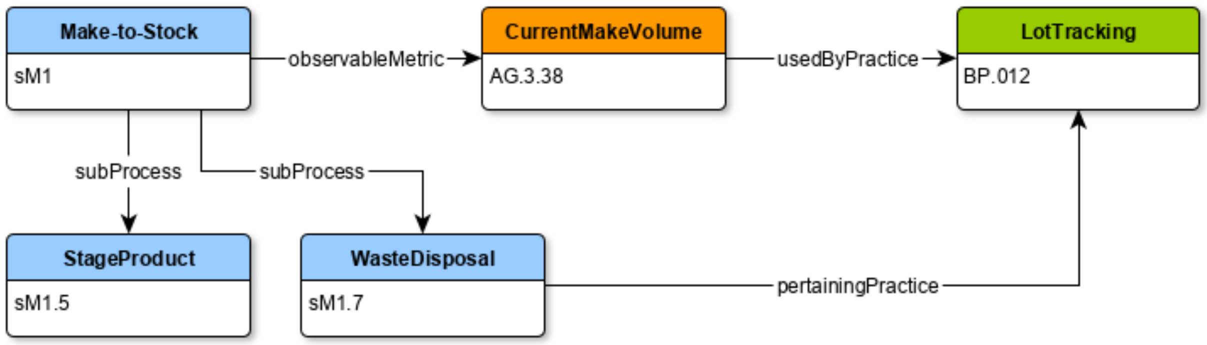


Figure 1: A small example of a graph

In this example, five triples create a simple graph further describing Make-to-Stock, including some of its subprocesses, the statement above and how Current Make Volume is linked to the Practice Lot Tracking (BP.012). Using URIs and a machine-readable syntax to express the triples involved makes the knowledge expressed with that graph accessible and computable by computers.

The Linked Data technology stack depends on three standards recommended by the World Wide Web Consortium (W3C)[1], namely

[1]Berners-Lee, Tim, James Hendler and Ora Lassila. "The SemanticWeb: A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities". Scientific American, May 2001. www.sciam.com/article.cfm?id=the-semantic-web%5C&%5C#38;print=true.

1. Extensible Markup Language (XML)[2], the basic data representation language of the World Wide Web
2. Resource Description Framework (RDF)[3], a foundational technology of the Semantic Web that is used to describe any concept or thing, such as a resource, in a machine-readable manner and serialize knowledge described with triples.
3. Web Ontology Language (OWL)[4], a description-logic-based language, expressed with RDF, that is designed to represent rich and complex knowledge about things and the relations between or among them.

OWL could be summarized as an ontology for defining ontologies. An ontology subsumes other commonly known knowledge formalizations, such as a taxonomy, schema, thesaurus, domain or information model, into one resource. Ontologies are not limited to these basic expressions of class hierarchies, formal naming and other simple relations. Instead, they are composed of axioms that constrain the possible interpretations of the defined concepts and relations to provide real semantics. Formerly implicit domain knowledge is thereby explicitly stated within an ontology.

For SCOR's purposes, an ontology is a set of concepts, properties and logical axioms with which to model a domain of knowledge or discourse. This conceptualization of a given domain or idea allows for a schema-bound representation of data with RDF and other formats and its interpretation by both humans and machines.

In addition to its formal characteristics, an ontology is useful as a central reference document for a given domain. A sufficiently documented information model, in concert with proper graphical representation, provides easy access to a complex domain.

Linked Data provides a universal, domain- and technology-independent way to portray resources in a non-proprietary format. It's extensible schema language, OWL, allows for use-case-specific consolidation and refinement of semantics, providing the syntax to define a bridge language for a given domain. Linking instances within a graph or to external resources is a powerful feature, providing additional context to strengthen or clarify the purpose of a given resource. The result is a web of data, a machine-readable, semantic network of structured data.

Note: In the context of this document, the terms ontology and information model are used synonymously.

[2] W3C. n.d. "Extensible Markup Language (XML)." Last modified October 11, 2016. www.w3.org/XML/.
[3] W3C. 2014. "Resource Description Framework (RDF)." Accessed 2019. www.w3.org/RDF/.
[4] W3C. 2012. "Web Ontology Language (OWL)." Accessed 2019. www.w3.org/OWL/.

SDSIM Architecture

This section will introduce two relational views of SDSIM architecture. The first depicts an abstract relational view across the entire value chain. The second depicts a relational view of the architecture to the SCOR Digital Standard.

Principles and general goals

Throughout the development of SDSIM architecture, the following principles were observed:

- Reuse existing, standardized, top-level ontologies. By providing a high-level association to standardized ontologies, we make it easier for new users to conceptualize SDSIM architecture, incorporate it into existing semantic environments, and construct their own interchangeable digital data objects.
- Operate under the Open World Assumption, which states that everything that is not explicitly defined otherwise could be true. This principle of formal systems of logic is the basis for the ease of extensibility and abstraction of OWL. It reduces the number of classes and relations defined explicitly while not obstructing future detailed specializations of an information model for particular use cases.
- Ensure extensibility. In order to successfully describe a domain with an ontology, a suitable extension of the provided axioms often is necessary to reflect the peculiarities of a given scenario. Extending an ontology has multiple advantages in contrast with creating one from scratch.[2] This approach is of particular interest to an abstract upper-level model, such as SDSIM architecture.
- Maintain interoperability between different domains, systems or data formats. This is a common requirement to enable data exchange with minimal loss of content, meaning or functionality. The interoperability between two models of the same domain tends to increase when the extensibility of at least one is heightened. An upper-level ontology can more easily interoperate with another ontology.[5]

The following general goals were used as a broad guideline:

- Model the value chain domain as described by ASCM specifications, primarily based on the current SCOR specification.
- Abstract all concepts and properties jointly used by all domains of the ASCM framework archives as a general upper-ontology, known as SDSIM architecture.
- Extend the domain description derived from the official ASCM documents with an enhanced view of metrics and a new event concept.
- Provide a structured foundation for any digital message exchanged between partners in a supply chain.
- Consider the added requirements for the complex and emerging challenges of the supply chain domain regarding its digital transition, particularly regarding advanced automation and maximum support for advanced analytics.

Namespaces

The namespaces used throughout this document and within the presented models are listed below. Each information model is associated with one unique namespace. Each entity defined within the model is appended to the pertaining namespace to create a globally unique identifier (URI), often dereferenceable by this URI in the World Wide Web. By this means, name conflicts are avoided, online documentation is linked directly, and the publishing organization is referenced by its domain. To reduce the length of URIs, prefixes are introduced to shorten the reused namespaces.

[5] For an extensive discussion on extensibility versus interoperability, see footnote 4.

Prefix	Namespace URI	Definition
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#	The RDF namespace
xsd	http://www.w3.org/2000/10/XMLSchema#	XML Schema namespace
owl	http://www.w3.org/2002/07/owl#	The OWL namespace
prov	http://www.w3.org/ns/prov#	The Provenance (PROV) namespace
skos	http://www.w3.org/2004/02/skos/core#	The Simple Knowledge Organization System (SKOS) namespace
SDSIM architecture	Forthcoming	The SDSIM architecture namespace
scor	**https://www.scor.ascm.org**	The SCOR namespace

For example, the concept Practice introduced with SDSIM architecture is identified by prepending the SDSIM architecture namespace to its name:

https://www.scor.ascm.org/domains/SDSIM architecture/Practice

This URI shows that Practice is part of the SDSIM architecture domain, which is under the authority of ASCM. Practitioners encountering this concept will be able to derive its meaning directly and can look up its exact specification under this URI. Any uncertainty about a concept or relation is thereby eliminated. Information models containing a concept of the same name will never be in conflict with it because it has a unique identifier. This identifier can be abbreviated by prepending the assigned prefix and a colon:

SCOR: Practice

Both expressions are semantically identical, provided the applied prefix was introduced beforehand. In the context of this document, however, the SDSIM architecture prefix will not be used in most cases, because almost everything that follows will be from the SDSIM architecture domain unless otherwise noted.

In the context of automatic processing of documents based on information models, such as SDSIM architecture, using unique identifiers is inescapable.

General disclaimer

The SDSIM architecture Information Model is finalized for its first version and is considered stable. The SCOR Information Model, based on SDSIM architecture, still is under construction and will be finalized in the future. This means that the names of classes, relations and attributes are provisional for now and still need to be evaluated by ASCM members. These elements were chosen to convey their semantic intent as closely as possible. In addition, the class structure and relations presented in this document could be changed or added to in the future, but any deletions should be minimal.

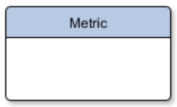
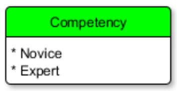
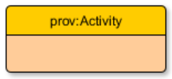
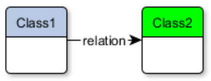
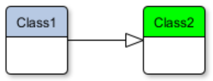
The majority of the SCOR model already is in place in the current information model, with the exception of the level-3 processes and metrics. This means that the current iteration of the information model is already usable.

The complete and detailed documentation for both information models, together with their machine-readable versions in RDF, will be released at a future date.

Note: The concepts and properties presented in this document are almost all defined for optional use. This means that none of the objects and relations described have to be used, depending on context and use case. For example, there is a property pointing out the start-event of a process, but a process-object does not have to provide this property in order to be considered complete in all cases.

Nomenclature and legend

In order to recognize basic semantics at a glance, the reader should be familiar with the terms and depictions presented in the below table.

Term	Depiction	Description
Ontology, vocabulary or information model	Depicted as a combination of the terms below.	A shared conceptualization and formal specification of a domain or idea. In this document, these words are used as synonyms.
Class or concept		A group of entities sharing an intrinsic and defining trait.
Enumeration		A special concept category containing a finite set of its instances, often predefined within the vocabulary and listed within the class depiction. Note that enumerations are denoted with an asterisk (*).
External class		A concept defined by an external vocabulary and always preceded by the appropriate prefix of said vocabulary.
Instance or object	Barring some exceptions, objects are not depicted within an ontology.	A member of the concept(s) it is instantiating. Often used as a representation of a real-world object, process or idea. For example, Contract Management (HS.0022) is an instance of concept Skill.
Relation or link		The manner in which two objects may be associated. Relations are defined on class level and have a unique name. These often are called links when relating external resources (similar to Web links).
Subclass relation		The transitive hierarchy relation connecting a subclass to a superclass. (Here Class1 is a subclass of Class2). This relation has its own representation without a name label.
Attribute	Label depicted within the class box.	A specific characteristic or quality of an object. Attributes relate literals often with the appropriate type. For example, “123”^^xsd:integer labels the integer 123.

In the remainder of this document, concepts and their instances will be referred to with capitalized names, italicized text and an optional leading prefix, to highlight that a given term has a formal definition within an information model. For example, *Metrics* refers to all known instances of the concept Metric. Relations and attributes are referred to by their names with an optional leading prefix, such as *involvedAgent* or *prov:generated*.

THE SCOR Digital Standard Information Model

1 SDSIM Architecture Overview Across the Value Chain

SDSIM architecture describes all jointly used concepts and interrelations of all members of the ASCM archived frameworks, (including the Product Life Cycle Operations Reference (PLCOR) model, Customer Chain Operations Reference (CCOR) model, and Design Chain Operations Reference (DCOR) model) as well as non-ASCM process models. As an upper-level ontology, SDSIM architecture is intended for reuse in all subdomain information models of the ASCM framework portfolio, which is demonstrated by its extension to form the SCOR Digital Standard. (See section 2) The benefits of this modularization approach are discussed in section 1.9.

Basing all information models of the ASCM framework portfolio on SDSIM architecture guarantees a large overlap of the concepts and relations in use. Thereby, an easy collaboration among different subdomains of ASCM is arranged and supported by the SDSIM.

1.1 Foundations

In an effort to align SDSIM architecture with existing and standardized information models, reflecting the essentials of this domain, the W3C Provenance Ontology was used as a foundation for SDSIM architecture. The Provenance Ontology (PROV-O)[6] is a widely adopted W3C recommended standard and serves as a lightweight process model. It expresses the provenance and interactions among activities, agents and entities.

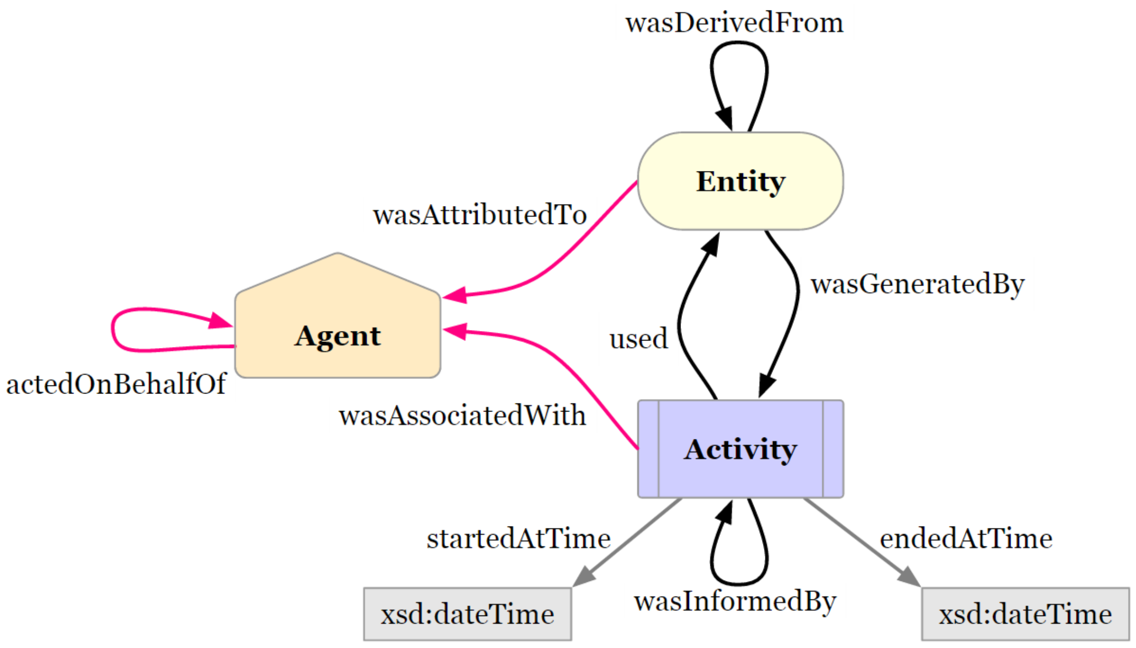


Figure 2: The core concepts of PROV-O and the relations connecting them[6]

It is crucial to provide a precise definition of these three core concepts because they are used throughout this document.

Entity

According to W3C, “An entity is a physical, digital, conceptual, or other kind of thing with some fixed aspects; entities may be real or imaginary”[6]. In other words, an entity is almost everything that is not classifiable as an agent or activity. In the context of SDSIM architecture, entities will subsume every physical object, such as materials and products exchanged; digital object, such as emails and files; and conceptual object, such as plans and practices. Furthermore, entities also are used to describe energy potentials, measurements and collections of entities. This extremely generic interpretation allows this concept to express all types of sustainability flow, including material, information and energy, along processes, relevant to the enterprise domain.

Activity

According to W3C, “An activity is something that occurs over a period of time and acts upon or with entities; it may include consuming, processing, transforming, modifying, relocating, using, or generating entities”[6]. Activities are the linchpin in every domain model because without them the described domain

[6] Lebo, Timothy, Satya Sahoo and Deborah McGuinness (Eds.). 2013. "PROV-O: The PROV Ontology." Accessed 2019. http://www.w3.org/TR/prov-o/.

would be static, dormant and without evolution. In the context of SDSIM architecture, Activities are equivalent to Processes. (See SDSIM architecture.1.8).

Agent

According to W3C, “An agent is something that bears some form of responsibility for an activity taking place, for the existence of an entity, or for another agent's activity”[6]. Common subgroups of agents include people, groups of people, organizations acting as a single agent, and software.

These concepts are the backbone of the W3C Provenance Ontology, onto which multiple serviceable concepts and relations are attached and reused throughout this information model. The complete interwoven fabric of SDSIM architecture and PROV-O will become evident at the end of this section.

1.2 Core Concepts

The following base concepts can be found at the heart of each domain reference model of the ASCM framework portfolio. (See Figure 3.) They constitute the cornerstone of an enterprise domain description and represent the main interface between the different frameworks supported by ASCM.

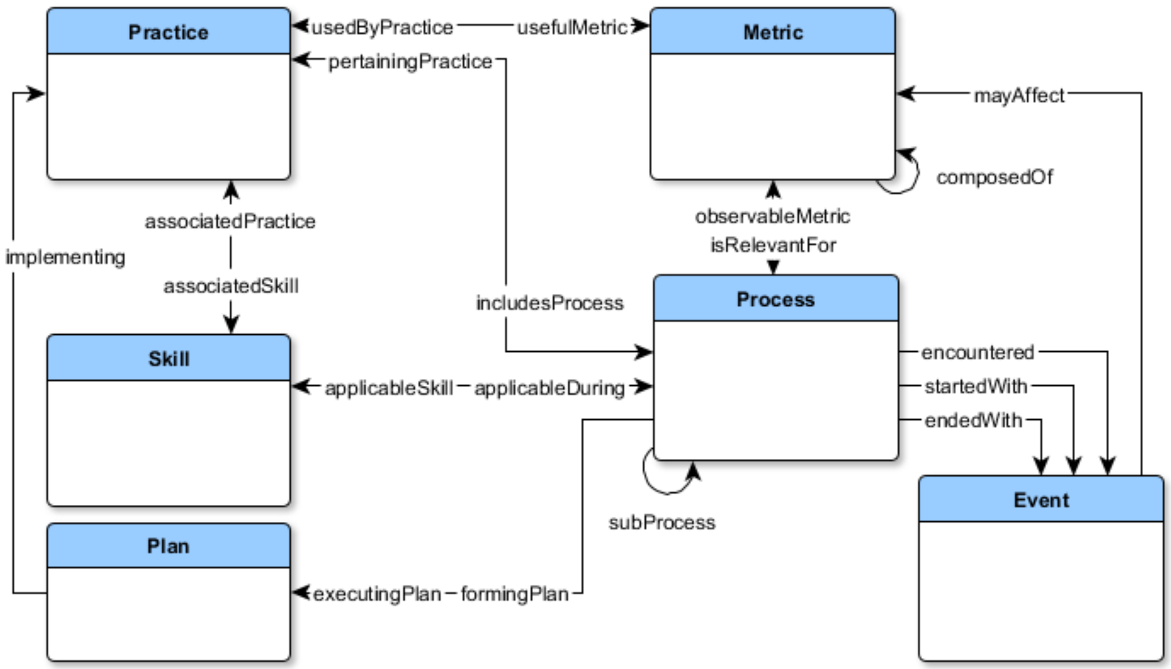


Figure 3: The core concepts of SDSIM architecture

Practices

A *Practice* is a unique way to configure a *Process* or a set of *Processes*. The uniqueness can be related to the automation of the *Process*, a technology applied in the *Process*, special *Skills* applied to the *Process*, a unique sequence for performing the *Process*, or a unique method for distributing and connecting *Processes* between organizations.

Skills

A *Skill* is the capacity of an agent to deliver predetermined results with minimal input of time and energy. The *Skill* of an agent and its proficiency often has a direct impact on the performance of an associated *Process*.

Processes

A *Process* is a unique activity that is performed to meet predefined outcomes and that occurs over a period of time. *Processes* consume or utilize entities to produce, modify or dissolve other entities that are exchanged between interdependent *Processes*.

Metrics

A *Metric* is a standard for measuring the performance of *Processes*. *Metrics* are categorized by a strategic direction of the performance evaluation for which they are employed, their scope and their purpose.

Events

An *Event* is an instantaneous occurrence or incident that marks a change to the environment described by the domain of discourse. Depending on the nature of the *Event*, one or multiple *Processes* and entities can be affected by its occurrence. Commonly, *Events* represent communications or interactions and are assumed to be instantaneous and atomic. *Events* can be encountered at any time during the duration of a *Process*.

Plans

A *Plan* is a conceptual entity that often represents a set of *Processes* as well as possible inputs and expected results. Depending on the context and domain, *Plans* can be created during a *Process*, while other *Processes* may execute a certain *Plan* to further their goals.

Note: Do not confuse this concept with the Plan Process in SCOR. In the context of SDSIM architecture, Plans are the result of such a planning process and usually are available as a printed or digital document.

Each of these concepts is surrounded by a number of related or dependent concepts, which are introduced in the following sections, along with all properties interlinking them.

1.3 Practices

In addition to the interrelations with classes *Process*, *Metric* and *Skill*, *Practices* are further qualified by the concepts *PracticeType* and *Policy*.

PracticeType

Practices can be grouped under this attribute, which describes the different types of *Practices* within any organization. It is important to understand that different *Practices* have different performance expectations, based on this attribute.

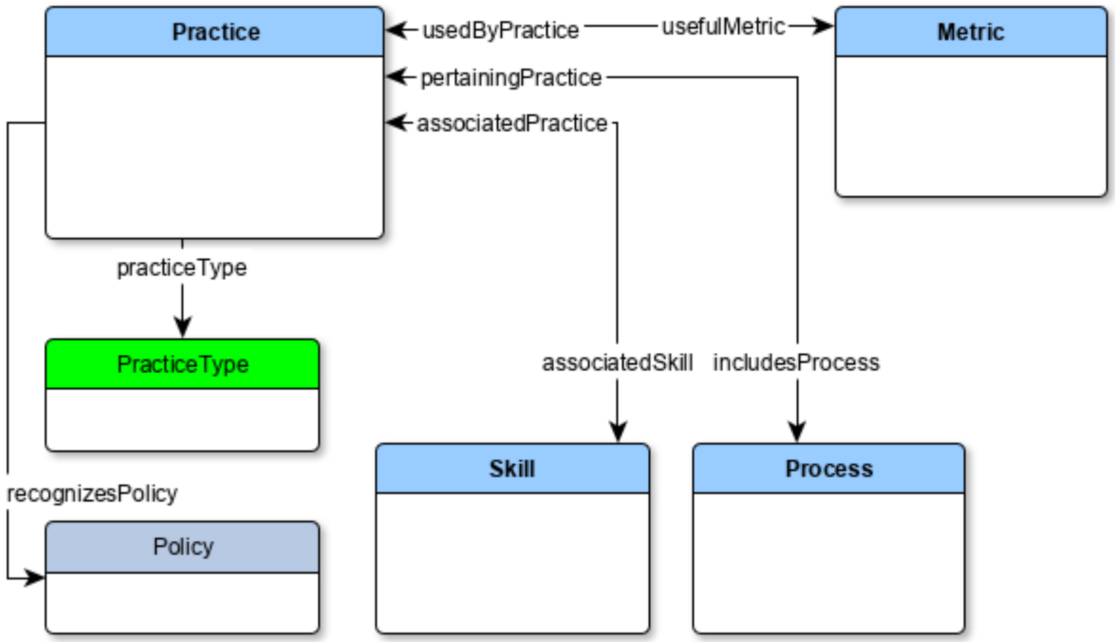


Figure 4: The *Practices* subdomain

Practices are described with the following relations:

- *usefulMetric* points out *Metrics* that often are used when validating the *Practice* and its associated *Processes.*
- *includesProcess* lists types of *Processes* that are a member of a typical implementation of the given *Practice.*
- *associatedSkill* represents *Skills* typically possessed by *Agents* involved in the included types of *Processes* of the given *Practice*.
- *practiceType* indicates the *PracticeType* a given *Practice* was categorized in.
- *recognizedPolicy* addresses known instances of a *Policy*, such as a regulation, statute, standard or guide, that might have an impact on the *Practice* at hand.

Practices can be implemented by *Plans.* (See section 1.5.). The *Maturity* of their mastery by an employee or organization can be evaluated with the concept *AgentAssessment.* (See section 1.4.)

1.4 Skills and Maturity

Skills are the center of a small landscape of concepts in the SDSIM architecture model. (See Figure 5.)

Agent

This is the adopted Agent concept of the W3C Provenance Ontology (PROV-O). In the context of SDSIM architecture, *Agents* represent organizations, people as members of organizations, and people as individuals. *Agents* may be attributed with a *Skill* (using *hasSkill*) to indicate some level of proficiency in said *Skill*. To further qualify this relation the *SkillProficiency* class was introduced. In addition, *Agents* may be assessed regarding their *Maturity* of command of a *Process* or *Practice*.

SkillProficiency

This qualification concept portrays the level of proficiency an *Agent* has regarding a single *Skill*. Using an instance of this concept and arranging it between an *Agent* and a *Skill* by the use of the properties *isProficient* and *inSkill* is equivalent to establishing a direct link between those concepts using *hasSkill*. However, by providing this additional mediator, it is possible to specify the level of proficiency by attaching a *SkillQualifiers*, using *experience*, *training* or *competency*.

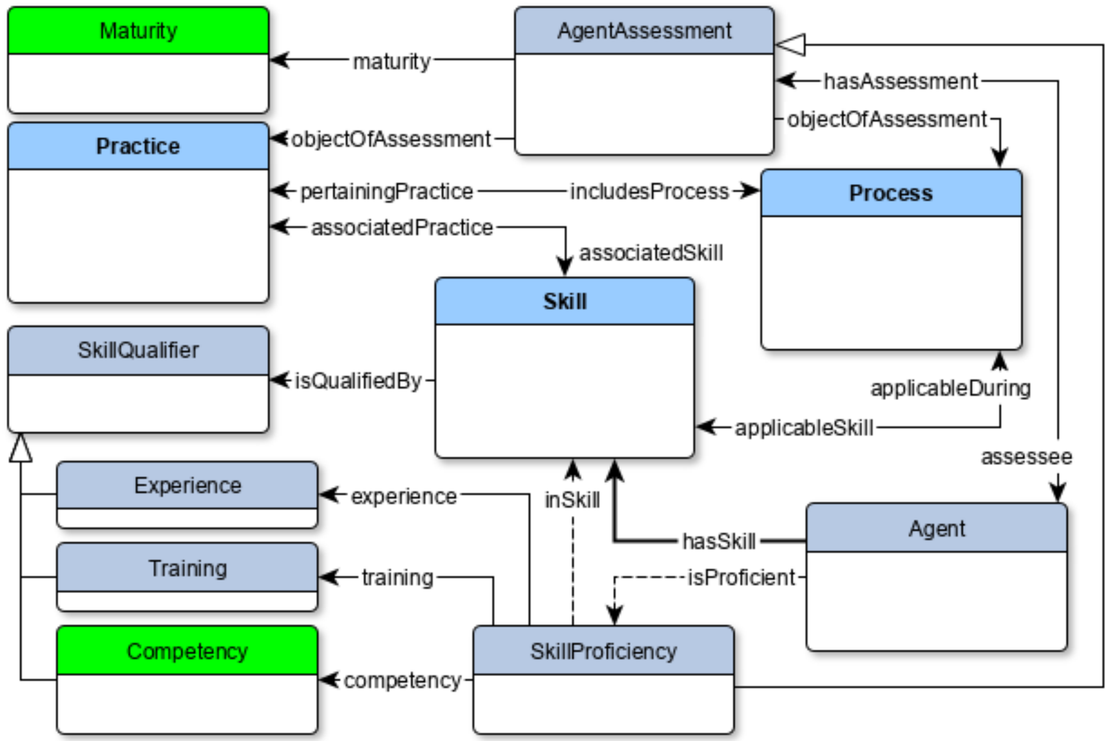


Figure 5: The Skills subdomain

SkillQualifier
This concept subsumes all instances of *Experience*, *Training* and *Competency*, which are qualifiers to define the level of proficiency of a *Skill*, under an abstract class defining their purpose as influencers of *Skills* an *Agent* may possess.

Experience

Experience is the knowledge or ability acquired by observation or active participation. *Experience* is obtained by doing the work in a real-life environment and undergoing different situations that require different actions.

Training

Training develops a skill or type of behavior through instruction.

Competency

Competency level describes the level or state of qualification to perform a certain role or tasks.

Skills are described with the following relations:

- *associatedPractice*, or the inverse property of *associatedSkill*, refers to *Practices* in which the given *Skill* might be of relevance.
- *applicableDuring* indicates those *Processes* that might be beneficially influenced by an involved *Agent* with this *Skill*.
- *isQualifiedBy* points out all instances of *SkillQualifier* (*Experience* or *Training*) that are relevant or appropriate as an indicator for a level of proficiency in this *Skill*.

The assessment of the *Maturity* of an employee, department or vendor regarding a *Process* or *Practice* is very similar to the qualification of *Skills*. In fact, the class *SkillProficiency* is just a specialization of the more general *AgentAssessment*, referenced with the property *hasAssessment* and its inverse property *assessee*.

AgentAssessment

A mediator for the assessment of *Agents* regarding a *Process* or *Practice* (or others, referred to by *objectOfAssessment)*. For example, the suppliers of a given product might be evaluated against an established maturity model[7] (using property *maturity*) regarding the *Process* Make-to-Order for the product at hand. This allows for the standardization of supplier evaluation. Additional reference concepts in addition to *Process* and *Practice* and additional qualifiers that complement the maturity evaluation also are conceivable.

Maturity

Maturity provides a qualitative comparison of supply chain *Processes* and *Practices* to descriptive representations of different levels of *Process* and *Practice* adoption and implementation. This evaluation measurement of supply chain *Process* and *Practice* effectiveness typically follows widely used models for practice maturity.

[7] Multiple industry and use-case-specific maturity models were defined by Gartner, such as the Five-Stage Maturity Model for Logistics Excellence, which can be found here: www.gartner.com/en/documents/3903399.

1.5 Plans and Involvements

This section introduces the environment of *Plans*, including the important qualification concept *Involvement* as a mediator between *Agent* and *Process*. (See Figure 6.) *Plans* usually are printed or digital summarizing a string of actions to be performed to achieve a certain goal. (Note: SDSIM architecture *Plans* are different from the SCOR Plan Process.) In the context of SDSIM architecture, *Plans* are separated into two subclasses: *EnterprisePlan* and *ExecutionPlan*.

EnterprisePlan

An *EnterprisePlan* defines and describes the collection of *Processes* required to align domain-specific *ExecutionPlans* and performance objectives with business strategies and goals as incorporated in the business plan.

ExecutionPlan

ExecutionPlans describes the committed plan for the execution of the company's activities, balancing the available resources and the aggregated customer demand. *ExecutionPlans* enable companies to institutionalize the management activities of their supply chains, resulting in a supply chain that better meets customer expectations and business goals.

Although *Processes* may implement the specification laid out by an *ExecutionPlan*, this achievement can only be accomplished through the actions of an *Agent* involved in the initiation, execution or finalization of the *Process*. In the case of automated processes, a software agent is used. The participation of an *Agent* in a *Process* is directly described with the property *involvedAgent* and its inverse *wasInvolvedIn*. The concept *Involvement* further qualifies this relation.

Involvement

An *Involvement* associates an *Agent* with a *Process* using properties *hadInvolvement* and *inProcess*, replacing the simple relation *wasInvolvedIn* with an explicit qualification concept. It can link the participating *Agent* with a particular role by using property *prov:hadRole* to indicate the *Agent's r*ole in regard to the executed *Process*. In addition, an *EnterprisePlan* or *ExecutionPlan* that was followed during the *Agent's* involvement can be attributed by property *prov:hadPlan*. Further qualifications, such as time frames or employed tools, can be added in sub-ontologies. (See section 1.9.) This approach follows the qualification pattern of *prov:Association*[8].

[8] W3C. 2013. "The PROV Namespace." W3C, May 19. www.w3.org/ns/prov#Association.

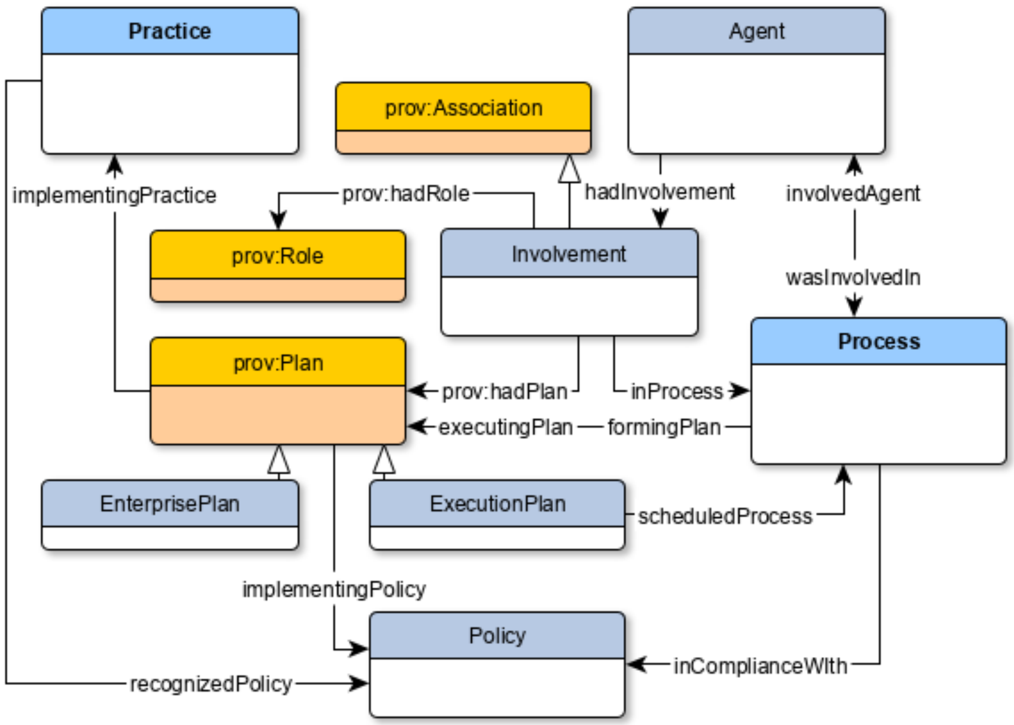


Figure 6: The Plans and Involvement section

Policy

A *Policy* is a formal system of rules describing duties, permissions and prohibitions to guide decisions and help achieve or prevent specific outcomes. *Policies* can be adopted by entire organizations, by subdivisions or in certain *Processes* through the relation *inComplianceWith* or implemented by *Plans* through the relation *implementingPolicy* for a specific use cases. Within SDSIM architecture, the concept *Policy* represents a wide range of possible subconcepts, differentiated by area of application and authority. These include laws and statutes; standards and technologies; business rules and procedures; as well as other binding and non-binding practices, conventions, stratagems, rules and guidelines.

At this level, *Plans* are described by the optional properties *implementingPractice*, which points out a *Practice* a *Plan* is trying to emulate, and *implementingPolicy*, which is used if Plans have to follow or incorporate a certain statute, standard or business rule. In addition, *ExecutionPlans* can define scheduled *Processes* using property *scheduledProcess*.

1.6 Metrics

In its essence, a *Metric* is a quantifiable measure of a certain dimension used to track and assess the status of a *Process* in regard to a strategic *PerformanceAttribute*. *Metrics* are structured hierarchically in all domains of the ASCM framework portfolio and can be differentiated by multiple facets: *PerformanceAttribute*, *MetricScope*, *MetricType,* and whether or not they are calculated based on other Metrics.

Particular attention was placed on the *Metric* subdomain of SDSIM architecture, extending the contemporary specification of this concept within the ASCM reference models. (See Figure 7.)

Note: A specific distinction between the different levels of the metric hierarchy is not necessary because a *Metric's* position in the hierarchy is self-evident. Only the top-level *Metrics* are referred to as strategic metrics or key performance indicators.

PerformanceAttribute

A *PerformanceAttribute* is a grouping or categorization of *Metrics* used to express a specific strategy. An attribute itself cannot be measured; it is used to set a strategic direction. Each of these attributes may observe at most one *Dimension*, such as time or money, which implies that each associated *Metric* also observes this *Dimension*.

MetricScope

The class *MetricScope* was introduced as an additional way to group *Metrics*. This class can be used to restrict the range of a given *Metric* to a specific area, function or use case. It can be implemented as simple labels or as a basis for a taxonomy of scopes.

MetricType

Metrics are further divided into five groups based on their fundamental purpose or focus:

- Descriptive or activity metrics simply monitor how much of something has occurred. They only tend to be useful when provided as a rate and with sufficient context information.
- A performance metric provides an indication of how well a process or a function is operating. Inventory turns, accuracy and customer service levels are important indicators of how well a supply chain organization is working.
- A diagnostic metric helps to explain why a performance metric is the way it is. Many diagnostic metrics are quality metrics because they provide an indication of performance in relation to a standard.
- Strategic metrics are diagnostic metrics for the overall health of a supply chain. These metrics are key performance indicators, and they help establish realistic targets to support strategic directions.
- Analytics are the forward-looking counterpart of metrics, which tend to measure what happened or what currently is happening. By comparison, analytics try to evaluate what is likely to happen and identify potential action(s) to enable or prevent a likely event.

Note: These metric types are not mutually exclusive. A given metric could, for example, be categorized as both a performance metric and a strategic metric.

ComplexMetric

This subclass of *Metric* represents all *Metrics* that are computed based on a set of lower-tier *Metrics* and that cannot be directly measured. For example, all *Metrics* presented by the SCOR specification are *ComplexMetrics* and need to be calculated based on *Metrics* provided by the implementing organization. In essence, *ComplexMetrics* describe the algorithm or operation used to calculate a *PerformanceObservation* of a *ComplexMetric*, which can be described in a machine-readable syntax with the property *formula*. *ComplexMetrics* provide an exact description of their *calculation* and list all dependent Metrics needed for their computation with the relation *requires*.

Measurement

A singular *Measurement* is recorded for a given *Process* and features a single numeric value, which is labeled with the attribute *numericValue;* a unit, which is labeled as *unit;* and the pointer to the observed *Metric*. This class is further specialized by its subclasses *PerformanceObservation* and *Benchmark*.

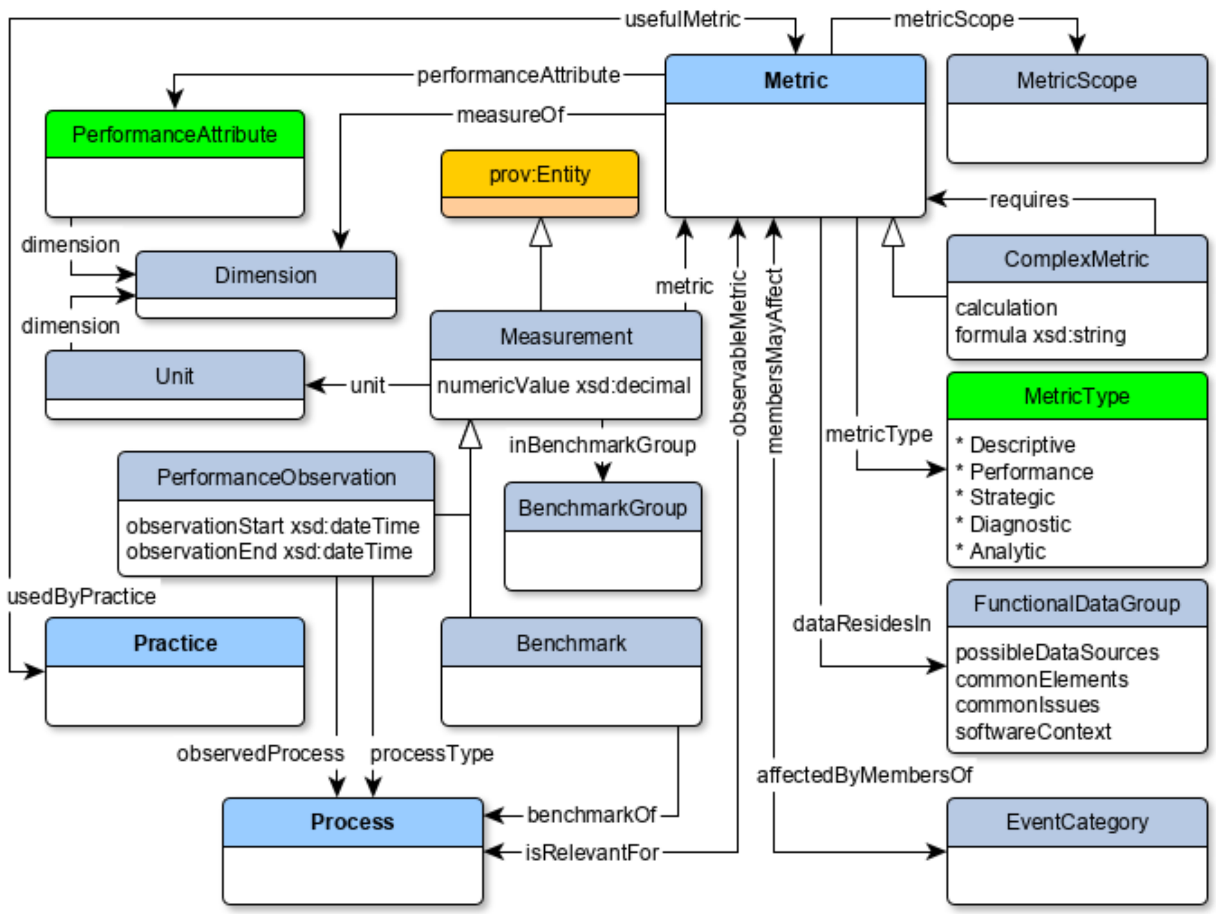


Figure 7: The Metrics subdomain

PerformanceObservation

This class represents a performance measurement for a specific *Process* type in regard to a single *Metric*. It has an optional time reference, denoted by the attributes *observationStart* and *observationEnd,* and specifies the observed *Process* type with the property *processType*. In addition, it can list all observed instances of a *Process* that influenced the observation, which are denoted with the property *observedProcess*.

Benchmark

Benchmarking is used as a guide to get a fact-based assessment of opportunities for improvement. A *Benchmark* serves as a reference measurement that *PerformanceObservations can be compared with to* gauge the performance of a *Process* against internal goals, competitors or external standards.

BenchmarkGroup

A benchmarking reference group usually describes a value range and gives a rough evaluation compared to or with a given scale or goal. Commonly, *BenchmarkGroups* are defined based on internal performance goals or as a comparison to peer groups, competitors or other third-party references. *PerformanceObservations* can be grouped using the class *BenchmarkGroup* and the property *inBenchmarkGroup*.

Unit and Dimension

Physical quantities are organized in a dimensional system built upon base quantities, which each have their own dimension and cannot be expressed with other quantities. This notion can be applied in other areas as well, such as finance. The *Unit* of a *Measurement* has to conform to the same *Dimension* as the *Metric* it is observing. The classes *Unit* and *Dimension* are only specified rudimentally within SDSIM architecture to satisfy the demands of defined Metrics. The notion of a dimension a unit is representing is universal, should apply to any unit system employed[9] to describe these semantics in detail, and can be expanded in sub-ontologies.

FunctionalDataGroup

This auxiliary concept was introduced to group required data elements by their characteristics, regarding their theme or hosting system, to provide implementing organizations with a lead as to where a required data element commonly resides. The included properties (*possibleDataSource, commonElement, commonIssues* and *softwareContext*) should aid in the search for the required data elements needed for the implementation of lower-tier *Metrics*. In this way, the effort to implement required, organization-specific *Metrics* should be reduced.

[9] Possible ontologies for this purpose include The Ontology of Units of Measure, which can be found here: enterpriseintegrationlab.github.io/icity/OM/doc/index-en.html, and NASA Quantity - Unit - Dimension - Type Ontology, which can be found here: data.qudt.org/qudt/owl/1.0.0/qudt/index.html.

1.7 Events

Events encountered during the execution of a *Process* may affect the *Process* and thereby its performance. While *Events* may be encountered at any point during a *Process*, a particular impact may derive from the *Events* associated with the *Process's* start and end. Therefore three properties refer *Processes* to the class *Event*:

- *encountered* refers to an *Event* that was encountered at any time during the execution of a *Process*.
- *startedWith*, a subproperty of *encountered*, refers to a *Start* Event that represents or coincides with the start of a *Process*.
- *endedWith*, another subproperty of *encountered*, refers to an *End* Event that represents or coincides with the end of a *Process*.

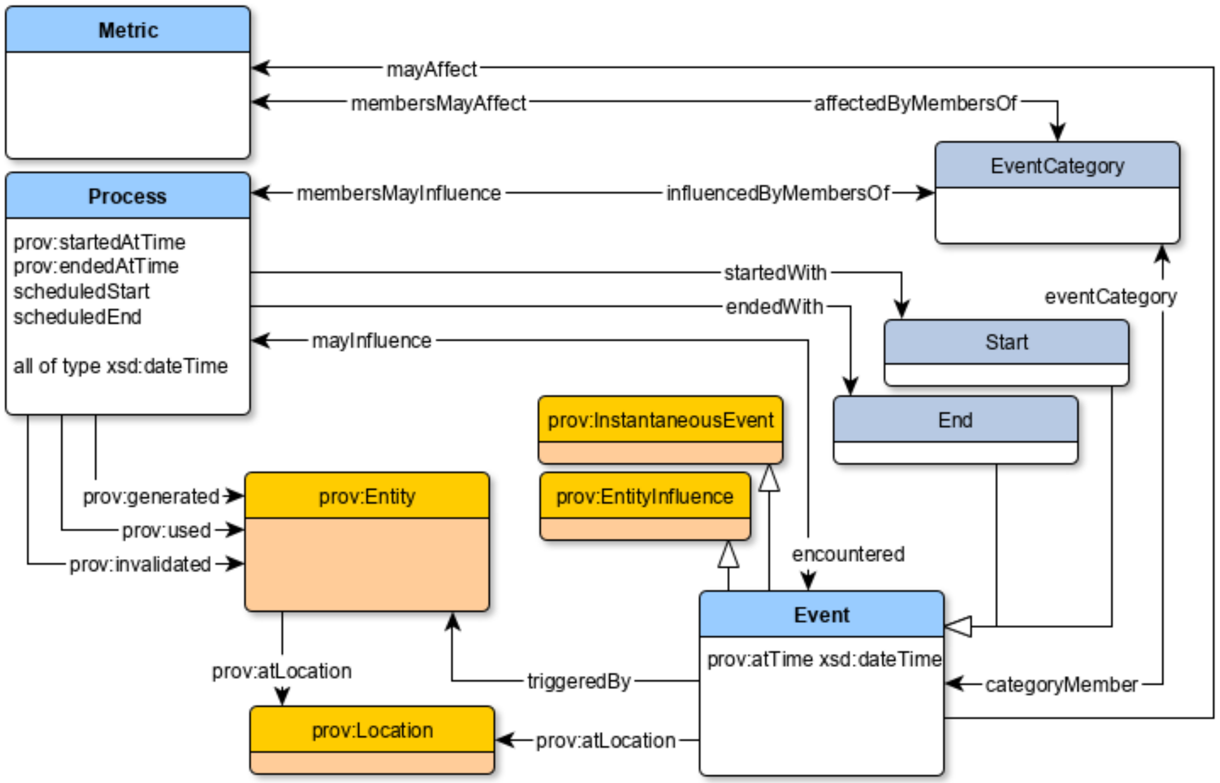


Figure 8: The Events subdomain

EventCategory

Events should be grouped under an *EventCategory*. Defining a particular *EventCategory* as a potential source for a *Metric's* performance changes by using the property *membersMayAffect* implies that each member of this group, which represents the actual events encountered, has a potential impact on the *Metric. Events* encountered by a *Process* may or may not have a direct influence on the *Process*. *EventCategories* with a heightened probability to influence a *Process* type can be described using the property *membersMayInfluence*.

Start and End

Both sub-concepts of *Event* highlight the importance of these specific *Events* for a *Process*. An instance of attribute *prov:atTime* has to align with the respective timestamp of the pertaining *Process* (*prov:startedAtTime* or *prov:endedAtTime*).

Events are further defined by their exact time of occurrence, which is represented using the property *prov:atTime;* an optional location, which is denoted by *prov:atLocation;* as well as a triggering entity, which is represented by *triggeredBy*, such as the arrival of an email. Because *Events* in SDSIM architecture are assumed to be atomic, complex events should be decomposed into their atomic parts or summarized and represented as a single atomic *Event*.

1.8 Processes

Processes are at the heart of the SDSIM architecture model and interconnected with all major concepts of this domain. (See Figure 9.) All related concepts were described in the previous pages. The class *Entity* from PROV-O is used in SDSIM architecture because it is highly generic and reusable in any conceivable use case. (See section 2.1). Because *Processes* are decomposable, the transitive property *subProcess* is used to point out all internal *Processes*, which together constitute a complex *Process*.

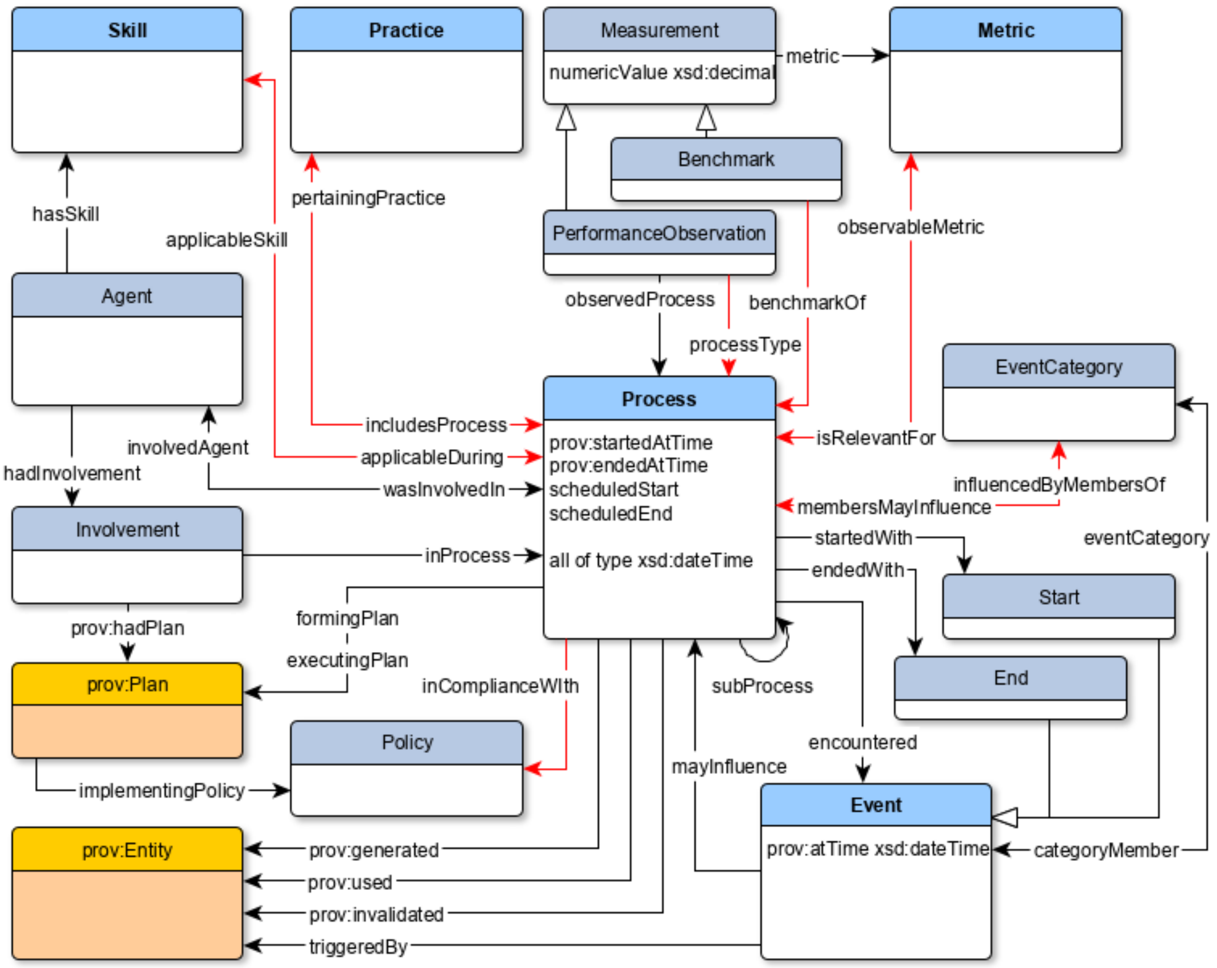


Figure 9: The Processes section at the center of SDSIM architecture

A distinction has to be made between properties addressing a *Process* as a type, which are colored in red in Figure 9, and those that attribute a single instance of a *Process*, which happen at a specific time and place[10]. For example, when defining a *Skill* as applicable for the *Process* Make-to-Stock (sM1), the SCOR specification declares that all occurrences of this process may benefit from *Agents* possessing said *Skill*. Then again, encountering a single *Event* at a certain time may only be relevant for a single instance of a *Process*. Because the term Process was conventionally used within SCOR to refer to whole classes of processes, the decision has been made to converge both concepts to prevent unnecessary confusion.

[10] This specific distinction is achieved at a formal level by introducing a subclass of *owl:Class* named *ProcessClass* and defining all subclasses of *Process* as an instance of *ProcessClass*.

Process types are described with the following relations and attributes:

- *pertainingPractice* is the inverse of *includesProcess* and refers to *Practices* a *Process* might be part of.
- *applicableSkill* is the inverse of *applicableDuring* and describes the set of *Skills* that is advantageous for *Agents* to possess when participating in the *Process.*
- *observableMetric* is the inverse of *isRelevantFor* and points out the *Metrics* that are relevant to observe in order to gauge the performance of a *Process* for the purpose of benchmarking or measuring progress toward a goal.
- *influencedByMembersOf* is the inverse of *membersMayInfluence* and indicates *EventCategories* that might influence the outcome of the *Process.* To be exact, their members, which are actual unique *Events*, may influence a given *Process* instance, which is expressed with the property *mayInfluence.*
- A *subProcess* is a component of a *Process. Processes* are decomposable into *subProcesses*.

Process instances are described with these properties:

- *formingPlan* and *executingPlan* both describe a *Plan* that is either established in the course of the current *Process* or executed, to achieve a goal set forth by the *Plan.*
- *involvedAgent* attributes some form of responsibility for the execution of this *Process* to an *Agent.* This property is further qualified by the class *Involvement.* (See section 1.5.).
- *encountered* refers to an *Event* that was encountered at any time during the execution of the *Process.*
- *startedWith*, a subproperty of *encountered*, refers to a *Start Event* that represents or coincides with the start of the *Process*.
- *scheduledStart* represents the scheduled start time of the *Process.*
- *scheduledEnd* represents the scheduled end time of the *Process.*
- *endedWith*, a subproperty of *encountered*, refers to an *End Event* that represents or coincides with the end of the *Process*.
- *inComplianceWith* indicates a *Policy* that was observed and adhered to during the execution of the *Process*. All requirements and restrictions defined by the given *Policy* should have been met.
- PROV-O properties *prov:generated, prov:used* and *prov:invalidated* provide entities involved in the *Process*, marking them as created, utilized or dismantled, or similar terms.

With these definitions in place, it is possible to formally describe the Process hierarchies of SCOR and other subdomains as defined by the SCOR specification.

1.9 Usage of the SDSIM

Finally, a complete conceptualization of SDSIM architecture is available. (See Figure 11.) Each of the enterprise domain frameworks of ASCM can reuse this abstract upper-ontology representing their shared domain features. The Web Ontology Language offers the means to import complete ontologies into dependent submodels. This approach will be demonstrated by conceptualizing the SDSIM in the next section. (See section 2.0) By the same means, further specializations for specific member groups, organizations or use cases are feasible and beneficial. The provided modularization of the ASCM domain allows users to choose the parts of ASCM domain that are useful for the given use case. This scaling approach adopts principles of the modular programming technique, separating concepts and properties of a large domain into independent, interchangeable modules that are specialized to fit common use cases and dependent only on SDSIM architecture. Figure 10 visualizes this approach by reference to an imaginary use case, appending the SCOR and DCOR models with additional semantics needed for this use case. Use-case-specific extensions usually include additional specialized concepts, relations and attributes.

In this instance, importing SCOR and DCOR into the use-case-specific extension model automatically imports SDSIM architecture as well. Other extensions of SDSIM architecture that are not directly imported, such as CCOR and PLCOR, are excluded from this use case, preventing unnecessary semantics to clutter up the final use case solution. Ontologies constructed in this fashion can provide the kernel of an enterprise knowledge graph. It specifies the semantics governing the graph and provides its first members.

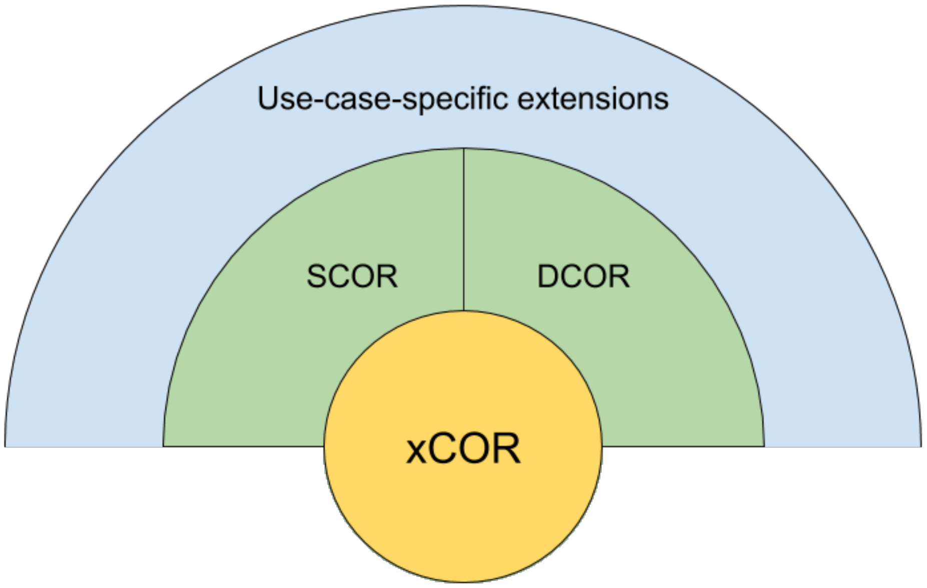


Figure 10: Composition of multiple ASCM models supporting a use case extension of the ASCM Information Model. The image represents a use-case-specific ontology that is based on SCOR and DCOR, which, in turn, are based on SDSIM architecture.
Credit: eccenca GmbH, Markus Freudenberg

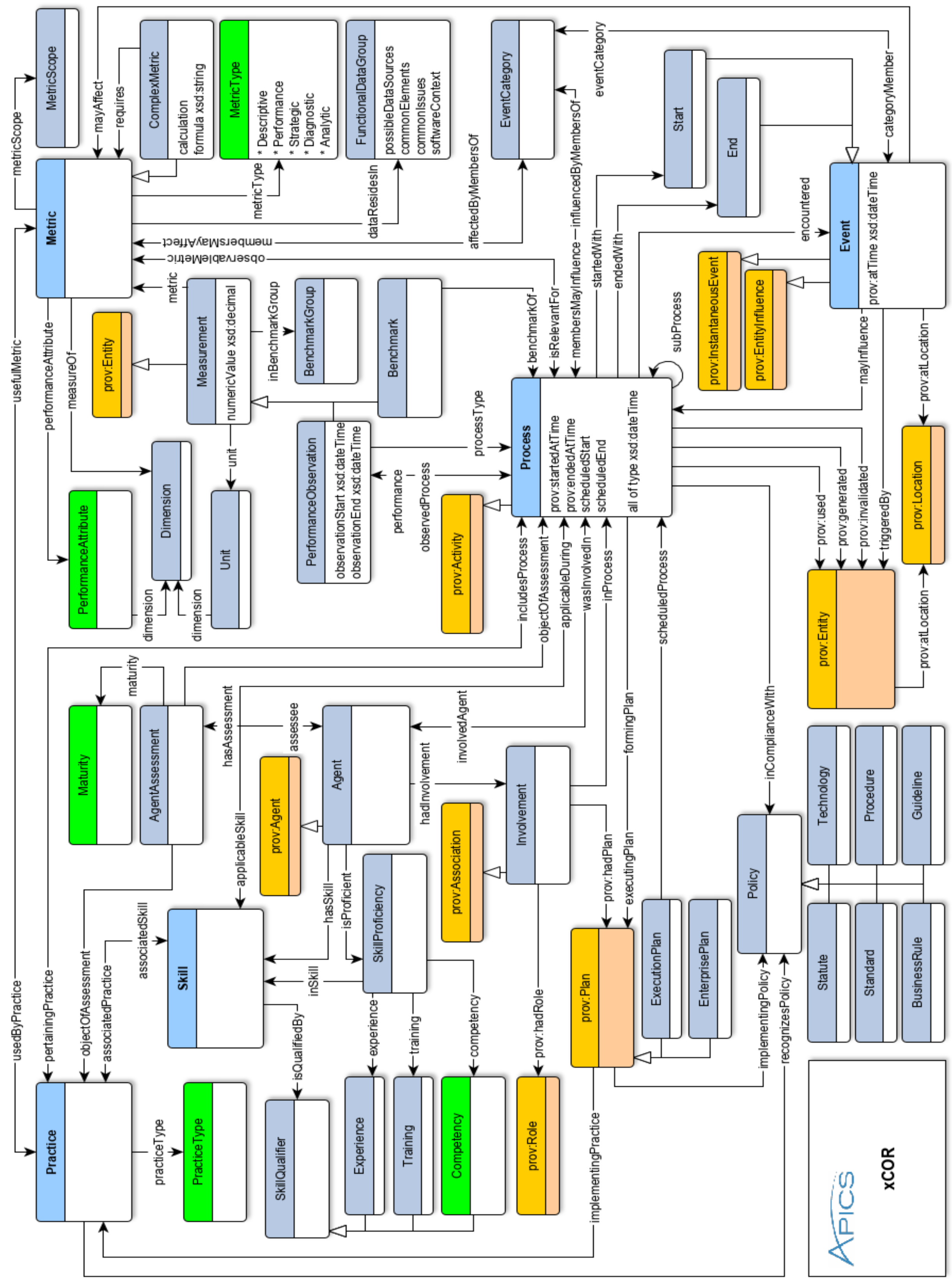


Figure 11: The SDSIM architecture

A minimal example of a use-case-specific extension could be the inclusion of additional *Skills* or organization-internal *Practices*, *Metrics* or *Processes* not listed in SCOR. In many cases, use cases require not just additional objects but additional semantics. For example, to further qualify the *Skill* of an employee, the highest educational degree may become relevant for some organizations. In order to model this specific requirement, the addition of a new subclass of *SkillQualifier* named Education and a property of the same name connecting it to the concept *SkillProficiency*, in combination with an enumeration listing possible degrees, could suffice. (See Figure 12.)

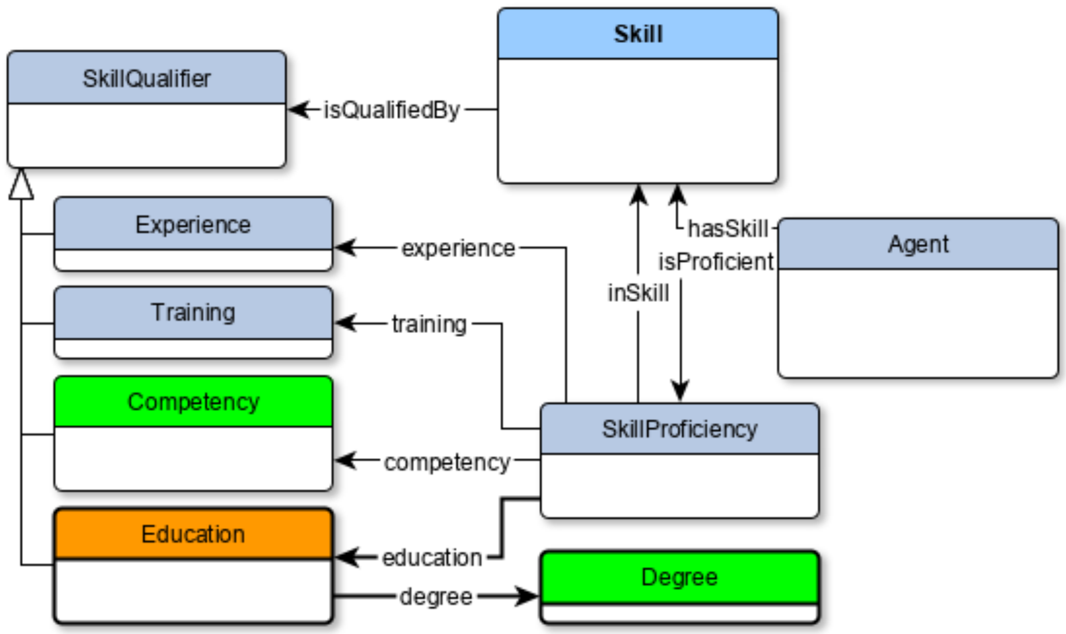


Figure 12: Example of a use case extension of SDSIM architecture

As is evident by its visualization in Figure 11, SDSIM architecture has been tightly interwoven with the Provenance Ontology by the W3C. Many SDSIM architecture concepts and properties were derived from general PROV-O classes and relations. In addition to its usefulness as a general process flow reference model, PROV-O enriches SDSIM architecture with its core objective. It “provides the foundation to implement provenance applications in different domains that can represent, exchange, and integrate provenance information generated in different systems and under different contexts”[6]. Provenance information, therefore, is an integral part of SDSIM architecture and is propagated along the string of processes of a supply chain. It is closely interconnected and can be enriched effortlessly by further metadata relevant to a use case. Such well-structured information is of great value for subsequent analysis tasks, such as demand forecasting, order-to-cash analysis and others.

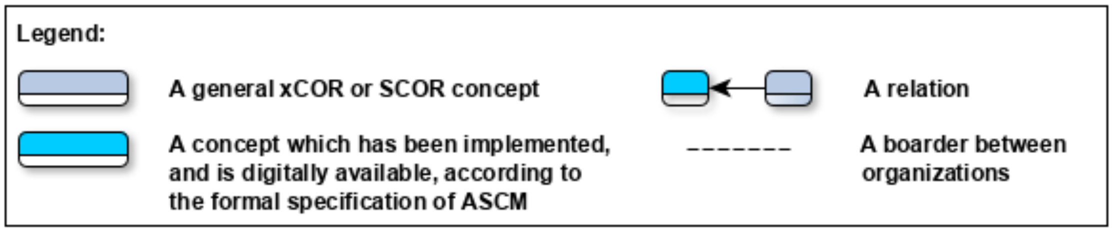
The SDSIM architecture is highly flexible, reusable and extensible and can easily interoperate with other formal specifications. It can be the basis for the digital exchange of data between supply chain partners, providing the highly visible information needed for crucial and early decisions. Furthermore, it provides a solid foundation for data-integration tasks, as detailed in the next section, and can act as the nucleus of an organization knowledge graph.

1.10 Data Integration

Note: This section has a more technical slant and is meant for tech-savvy practitioners and information technology (IT) professionals involved in data-integration endeavors. However, it is important for any supply chain practitioner to understand the importance of data integration to any data-centric solution, how costly it can be, and why a unified information model is critical for this effort.

The presented graph-based data model, while easy to comprehend, is often foreign to conventional information landscapes and the database-centric IT environments of organizations. Although the benefits of knowledge graphs at the center of the information ecosystem can be plentiful[11], the effort of establishing such a data layer can be a challenging exercise. This section will outline an incremental strategy for adopting this specific perspective on data, revolving around the omnipresent and important task of data integration. The four levels explained below represent the degree to which an organization is exploiting the benefits of this ontology-based approach to limit the costs of data integration as much as possible. Except for the first level, which is merely informative, data integration is a necessary process for all approaches presented here and for any data-centric solution in general.

Before any sophisticated analysis task can be undertaken, a sufficient amount of data has to be collected. Modern IT environments consist of many possible data sources, often using incompatible schemata. This raises the demand for the integration of the collected data under a common information model in order to achieve a unified view of all data. In fact, the time to extract, align and clean the input data is estimated to take up between 30% and 50% of the overall time needed to complete a given machine-learning task[12]. Because the demand for machine-learning and artificial intelligence solutions is constantly increasing, a reduction of this lead time is of primary concern. The complex task of data integration is one of the foremost objectives of enterprises seeking to expand their digital repertoires[13].


Legend:
A general xCOR or SCOR concept
A relation
A concept which has been implemented, and is digitally available, according to the formal specification of ASCM
A boarder between organizations

Level 1: Reference model

The ontology and its documentation, alongside official specification documents, are used as a detailed reference model of the domain. They help in identifying important relations or dependencies and support the efforts to refine diagnosis, benchmarking and the overall methodology of supply chain management.

[11] Mikhail Galkin, Mikhail, Sören Auer and Simon Scerri. 2016. “Enterprise Knowledge Graphs: A Backbone of Linked Enterprise Data.” 497-502.
[12] Sapp, Carlton E. 2017. “Preparing and Architecting for Machine Learning.” Technical Professional Advise, Gartner Institute.
[13] Based on the top response (44%) in a recent survey of 1,400 executives released by Progress: www.progress.com/campaigns/datadirect/whitepapers/2018-data-connectivity-annual-report.

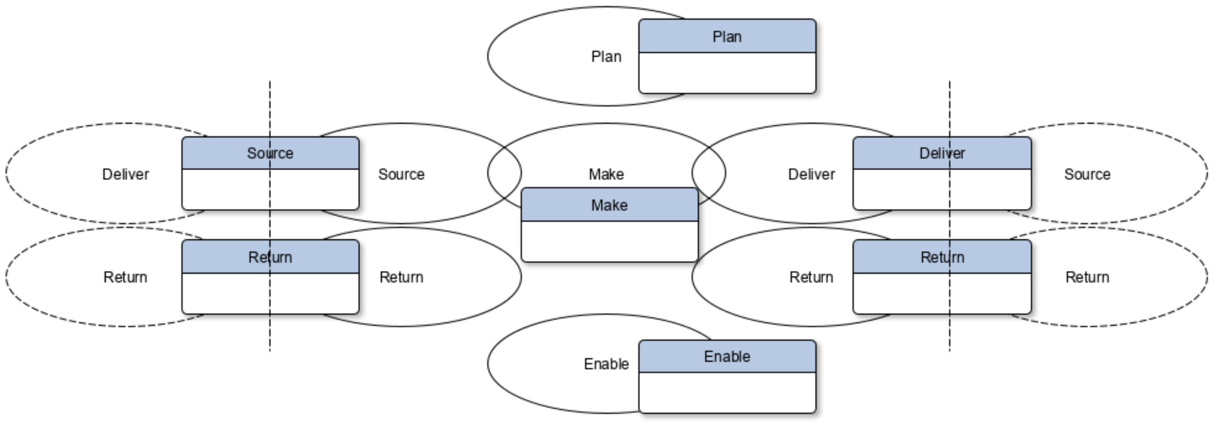


Figure 13: In Level 1, knowledge of concepts and relations is deepened.
Credit: eccenca GmbH, Markus Freudenberg

Level 2: Standardized interfaces

At this level, an organization opts to implement its data interfaces with supply chain partners following the specifications set forth by ASCM. Through this effort, the organization provides highly visible, real-time information to its direct supply chain partners about current market interactions, material flows, product and design specifications, inventory and process capacities, or provenance and metadata. Standardized interfaces are the basis for any effort to interlink supply chains over disparate industries with different enterprise software solutions that use different languages or terminology. Because the digital objects exchanged between partners are directly derived from the ASCM information models, a precise understanding of the introduced concepts and relations is paramount.

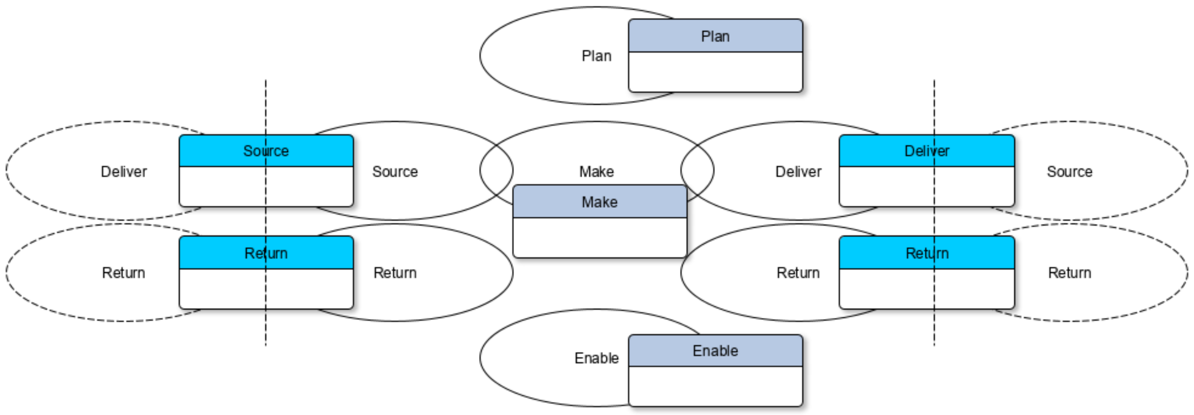


Figure 14: In Level 2, communication between organizations is machine-readable and standardized. However, only objects commonly exchanged between the partners need to be implemented according to the model.
Credit: eccenca GmbH, Markus Freudenberg

Level 3: Integrated data sources

At this stage, SDSIM architecture and other ASCM information models are an integral part of the internal data landscape of an organization, usually extended with organization-specific semantics. Data sources, such as relational databases, are mapped against these models to establish a unified and graph-based view of the data. This is necessary because diverse and distributed IT solutions usually are built on separate databases using different, incompatible schemata.

The third stage is accomplished by a non-disruptive data-integration process that establishes automated data workflows to extract and transform the origin data objects into members of the concepts defined above. Following this approach inherently provides Level-2 interface compatibility. Integrated data sources are employed to gain organization-wide insights, to interlink entities of disparate data sources by breaking open data silos, and to enrich the data with formal domain knowledge.

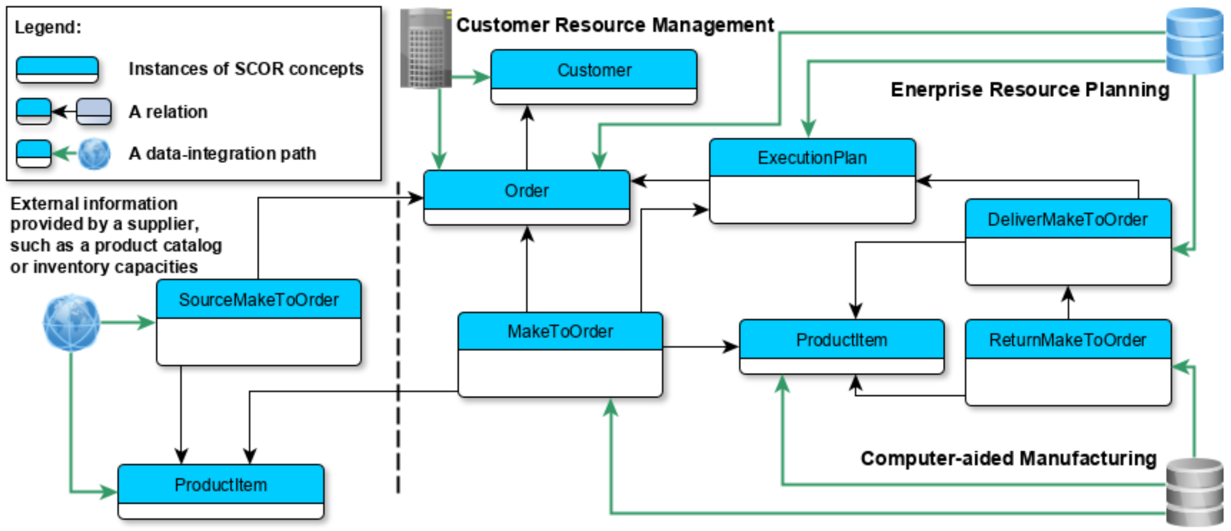

Legend:
Instances of SCOR concepts
A relation
A data-integration path
External information provided by a supplier, such as a product catalog or inventory capacities
Customer Resource Management
Enerprise Resource Planning
Computer-aided Manufacturing
Customer
Order
ExecutionPlan
DeliverMakeToOrder
SourceMakeToOrder
MakeToOrder
ProductItem
ReturnMakeToOrder
ProductItem

Figure 15: In Level 3, internal and external data sources are integrated under an SDSIM architecture-based information model.
Credit: eccenca GmbH, Markus Freudenberg

Level 4: Knowledge graph

At this stage, many of the conventional data sources are synchronized with graph-based solutions, such as triplestores, property graphs and others, feeding a central, materialized knowledge graph. Instances of SDSIM architecture concepts are first-class citizens of this data landscape, in contrast with the previous stage. A central and up-to-date knowledge graph eliminates most of the data-integration tasks, such as extraction and mapping workflows, further reducing the lead time for subsequent analysis tasks. Establishing such a data structure at the center of an organization’s information landscape should not be considered the final and ultimate goal. Although it is helpful in many ways, the return on investment for this effort depends on multiple factors, such as the number of related projects, the additional tools required, capabilities and capacities of the involved IT professionals, the number of involved information systems and others. Often, a synchronized data-integration effort of all data-centric solutions (such as Level 3), which helps avoid having incompatible information models used for different tasks in one organization, is the best strategy for large companies with established IT environments.

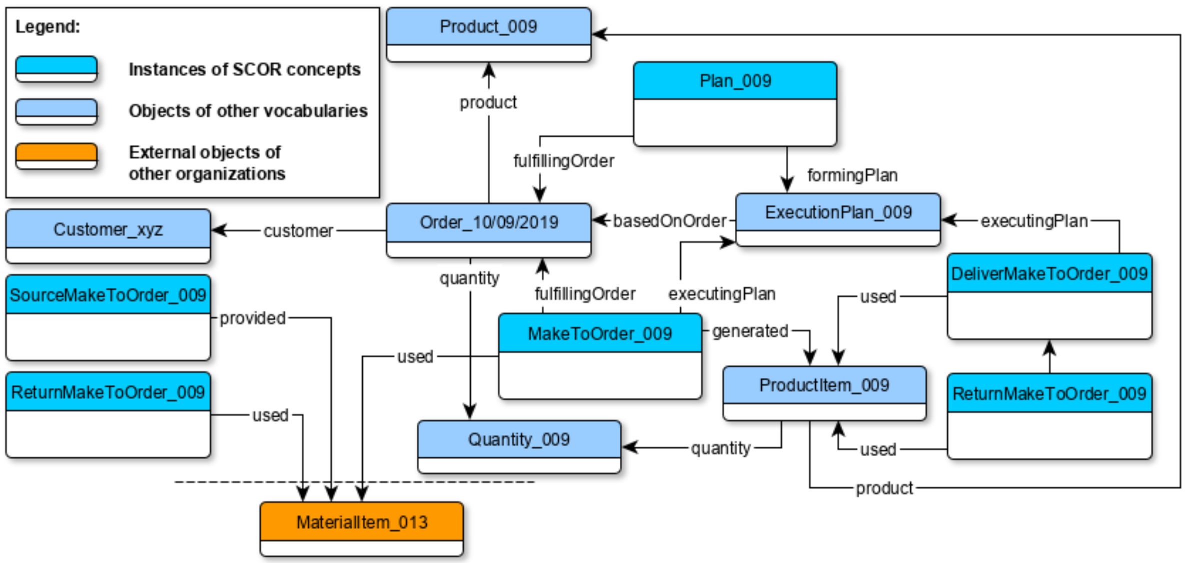


Figure 16: A centralized knowledge graph covering the many data sources of an organization is a key component of Level 4.
Credit: eccenca GmbH, Markus Freudenberg

Data integration is a common, even unavoidable, preparation task when working with data in most organizations. Having an established and unified domain or organization-wide information model reduces this effort significantly. Reusing this model for each analysis task diminishes the data-modeling exercises to simple use-case-specific model extensions, as described above. (See section 1.9.) Mappings of specific data sources to the established information model and pertaining workflows can be repurposed to a large extent. In addition, changes to the domain model can be performed in a centralized way, reducing the amount of additional work needed to adapt existing data-integration workflows and diminishing the error risk.

Furthermore, the Linked Data approach, implied by this ontology-centric data integration, offers additional benefits:

- Globally unique identifiers, which are by their nature unambiguous, can be used in any system without fear of conflicting with other identifiers, such as those used by partner organizations.
- URIs eliminate ambiguity and provide the basis for Linked Data.
- The complete Linked Data technology stack is non-proprietary and can be used independently of specific tools or domains. All described technologies are standardized by the W3C and are openly available and free to use.
- Links between data objects enrich the information contained in a given graph by pointing to a more elaborate description located in another graph or dataset or by providing additional context in general. For example, using the digital product specifications of a partner organization by linking to the appropriate URI renders a local copy of it redundant. Exchanging a single URI provides all context necessary to distinguish a product from any other, as well as providing its exact specification at the web location of the URI. (This is optional, but highly recommended.)
- The veracity of data, encompassing data quality and trustworthiness of its source, is a highly active field of research.[14] The importance of data veracity, especially for subsequent analysis tasks, is crucial and should not be ignored. A high-level vocabulary for defining quality requirements — independent of

[14] Common issues include biases, missing provenance, abnormalities and noise, precision issues, and outdated measures. This increases the uncertainty in the source and adds to data quality issues.

implementation, technology and domain — has been developed and standardized by the W3C.[15] Additionally, proprietary and open-source validation engines already are available.[16] They can validate entire graphs for adherence to information models and additional quality demands.

A Linked Data approach for the integration of multiple data sources suggests that the final outcome is data in an RDF format. However, any syntax or format is conceivable. While using a Linked Data methodology for the purpose of integration, the approach has no restriction on the desired format. Therefore, such data-integration workflows can be incorporated directly into any machine-learning, analysis or other data-provisioning workflows.

[15] W3C. 2017. "Shapes Constraint Language (SHACL)." July 20. www.w3.org/TR/shacl/.
[16] For example: Topquadrant SHACL validation, which can be found here: www.topquadrant.com/technology/shacl/, or RDFUnit, which can be found here: https://github.com/AKSW/RDFUnit.

1.11 Competency Questions and Evaluation

The development of an information model usually is driven by a set of competency questions. These questions define the objectives, scope and expressiveness requirements of the domain. They are used to evaluate the conceptualized model, to ensure conformity to expectations or to show limits of its use. The following list of competency questions was collected at the SCOR model update meeting in Chicago in April 2019. The questions focus in particular on complex and emerging challenges of the supply chain domain.

This section will evaluate the SDSIM architecture Information Model by answering these competency questions, summarizing the intended use of SDSIM architecture. It will demonstrate how the SDSIM architecture model is able to satisfy many general requirements of the ASCM domains and how SDSIM architecture supports the complex, digital challenges of the supply chain domain.

Note: The answers provided for each question are written on a general level to showcase capabilities and opportunities of the SDSIM architecture Information Model. The competency questions are discussed on a more detailed and technical level in the SDSIM architecture online documentation, which will be available at a future date. roups A and B demonstrate SDSIM architecture's integration into the information landscape of an organization, and Groups C, D and E describe SDSIM architecture's use for common problems regarding the domain's digital transition.

A. General application in the ASCM framework portfolio

1. **Can we model organization-internal Processes?**
   New *Process* types can be declared for any level of the domain process hierarchy, either as a new top-level *Process,* such as SCOR Enable, or as a subclass of an existing *Process* class. In the context of SDSIM architecture, additional *Process* classes usually are defined as organization-specific subprocesses of SDSIM architecture *Processes*.
2. **Can SDSIM architecture portray relevant technologies, policies and standards?**
   The concept *Policy* was introduced specifically to cover this requirement. It includes all types of binding and non-binding regulations, procedures and guidelines and is referenceable by *Practices*, *Plans* and *Processes*.
3. **How can we describe the skills of an employee, as a basis for an automated ticket allocation, using SDSIM architecture?**
   The level of proficiency regarding a single *Skill* of an employee is qualified by the concept *SkillProficiency*. Instances of *SkillQualifier* include *Experience*, *Training* and *Competency* and allow for the attribution of relevant qualifications defining the proficiency of the employee. The description of *Agents* and their *Skills* can be extended, for example, through an organization-wide SDSIM architecture extension, to portray other important qualities necessary to satisfy the ticket-allocation use case. (See section 1.9.)
4. **Can SDSIM architecture represent every type of sustainability flow, including material, energy, money and others?**
   Yes, the concept Entity of PROV-O, which is used throughout SDSIM architecture as a conceptual stand-in for any type of input or output for *Processes*, is generic enough to represent all sustainability flows. However, for each of these use cases, an extended information model is advisable to cover all peculiarities.

5. **How can we associate a particular Practice to a Process that followed its recommendations?**
   Although the property *pertainingPractice* exists to link a *Process* type and a certain *Practice* directly at a generic level, it does not specify which particular *Practice* was used while a *Process* was executed. (See sections 1.3. and 1.8.) To achieve this, the concept *Involvement* has to be used. *Involvements* of *Agents* can define *Plans* that were observed by the involved *Agent*, which in turn can specify a *Practice* implemented by the associated *Plan*.
6. **How does SDSIM architecture define an ExecutionPlan in order to schedule a chain of Processes?**
   Although the class *ExecutionPlan* is quite simplistic at the SDSIM architecture level, scheduling *Processes* is possible by using the property *scheduledProcess*. This relation may list all scheduled *Processes* for a given scenario in order to achieve the goal of the execution phase. In addition, the attributes *scheduledStart* and *scheduledEnd* were introduced to schedule a timeframe for a *Process's* execution.

B. Application in Process-performance measurement

1. **Can we gauge the performance of *Processes* and compare them to a set of *Benchmarks*?**
   The performance of a *Process* can be recorded for a given *Metric* using the *PerformanceObservation* concept. (See section SDSIM architecture 1.6) A comparison against *Benchmarks* can serve two purposes: to first compare the attributed *Units* of each *Measurement*, followed by an optional conversion of values, and then to compare the *numericValue* attribute. The property *inBenchmarkGroup* can be used to assign a given *PerformanceObservation* to a value range, defined by an instance of *BenchmarkGroup*, based on such a comparison.
2. **Can we identify the data necessary to create a Perfect Order Fulfilment *Metric* in SCOR?**
   To satisfy the demands of a *Metric* defined, for example, by SCOR, specific input data has to be provided or combined into organization-specific *Metrics*, which feed upper-level SCOR *Metrics*. The type of data necessary is described by the concept *FunctionalDataGroup*, which points out typical locations, example data and common issues. It should help to minimize these efforts and reduce ambiguity.
3. **Can we organize Metrics in a taxonomic system and add additional facets to structure the multitude of organization-internal metrics?**
   Metrics defined by ASCM frameworks, such as SCOR, are structured as a taxonomy. Additional *Metrics* can be included in this taxonomy. In addition, the concept *MetricScope* was introduced to provide non-hierarchical categorizing of *Metrics* (also called tagging) or to restrict the usage scope of the *Metric* in question.
4. **Can we define use-case-specific *Metric* definitions as a composition of other *Metrics*?**
   Key performance indicators and other high-level *Metrics* are composed of multiple lower-tier *Metrics*. The exact relation and calculation of such *Metrics* can be defined with SDSIM architecture, providing an exact account of all necessary input data. (See section 1.6.)

C. Application in advanced automation

1. **Can SDSIM architecture accurately portray any concept of a supply chain environment, as a basis for automation tasks?**
   Basing SDSIM architecture on PROV-O and reusing its three core classes, *Entity*, *Activity* and *Agent*, allow for a highly generic description of any domain. (See section1.1.) The specialization of these semantics to describe a use case scenario can, therefore, reuse these generic concepts and others to define any object, process, event, idea, plan, organization, team, location, measurement or other object at any level of detail.
2. **Can we enforce a general digital fidelity in order to avoid automation based on incomplete or erroneous data or processes?**
   The term digital fidelity refers to the strict adherence of digital objects to a given schema, such as a vocabulary or ontology. This is important when exchanging such objects with other organizations or members within an organization to avoid ambiguity or unknown semantics and ensure that the communicating partners understand each other. Such adherence can be enforced when exchanging messages by validating a given digital message. Tools and standards necessary for such data validation exist (See section 1.10) and can be employed to ensure correctness and completeness of a digital object. Describing a *Process* and its outcome digitally allows for the validation of its digital representation and thereby the validation of the process itself.
3. **Can SDSIM architecture keep track of every *Entity, Process* or *Agent* involved in a supply chain, regardless of organizational borders, thereby establishing provenance?**
   Establishing provenance records, independent of domain and context, is a declared goal of PROV-O, which is tightly interwoven with SDSIM architecture. The direct links between objects based on properties defined by PROV-O help to navigate the source of a data point. Furthermore, establishing standardized data interfaces between supply chain partners will allow extending this graph to include provenance data of customers and suppliers, particularly about products, supplied goods, stock and capacities.

D. Application in advanced analytics

Descriptive analytics:

1. **Can SDSIM architecture provide a basis to define data for relevant use cases?**
   As an upper-level ontology, SDSIM architecture was designed to be as generic as possible. This implies its ease of extensibility and good interoperability with other formal specifications, as discussed in section 1.2. The SDSIM architecture model focuses on domain processes supporting a sustainability flow. It enables a wide range of use cases in an enterprise domain, as is evident by its applicability to the entire ASCM framework portfolio.
2. **Can SDSIM architecture incorporate external vocabularies, ontologies and taxonomies for reuse, in order to avoid reimplementation of existing solutions?**
   The extensibility and interoperability of SDSIM architecture not only allows for specific use case extensions, but it also supports the import of external information models. This further reduces the effort of extending SDSIM architecture. Additionally, throughout Linked Data ontologies, PROV-O is used as a process reference model[17] by aligning with its concepts or by import directly. This allows SDSIM architecture, as a sub-ontology of PROV-O, to interact effortlessly with such models.
3. **Is SDSIM architecture useful as a central reference model for data integration tasks and replacing master data management?**
   As described in section 10, SDSIM architecture is very well suited to act as a semantic nucleus for an ontology-based data-integration effort. Typical issues with poor master data management can be

[17] Such as the W3C Semantic Sensor Ontology for sensor data, which has a direct alignment with PROV-O: www.w3.org/TR/vocab-ssn/#PROV_Alignment.

avoided because of the inherent features of the data model, such as the use of URIs and enhanced data context through links and relations. In addition, the focus on provenance, by using PROV-O, is helpful when disambiguating possible duplicates.

Predictive analytics:

4. **Can SDSIM architecture help with predictive analysis and machine learning?**
   Data-integration tasks can account for as much as 50% of the overall time needed to implement a machine-learning task. The SDSIM architecture model can help to reduce this lead time significantly, when used as a basis for data integration as described in detail in section 1.10. The centralization of data-integration efforts over multiple projects benefits greatly from a unified information model and is of high importance, especially in larger organizations. In addition, rich provenance information and digital fidelity to the schemata used for the machine-learning task, as well as high data quality, are benefits not to be underestimated.
5. **Can SDSIM architecture increase data quality in order to reduce data noisiness and increase pattern detection accuracy?**
   Including automatic data quality validation into the data-integration workflows contributes significantly toward the veracity of the collected data[18] and ensures that important analysis tasks are not executed on incomplete or erroneous data. As an ontology, SDSIM architecture provides a basic formal definition of the value chain domains, containing many constraints against which data objects can be validated, such as the domain and range of properties, hierarchical constraints, data type validity and others. Data tests for validating such constraints can be derived automatically from SDSIM architecture by using state-of-the-art data validation tools. (See section 1.10.) Further tests can be defined directly. By these means, input data is automatically sifted to ensure completeness as well as syntactic and semantic correctness.

Prescriptive analytics:

6. **Can SDSIM architecture be utilized as a foundation for simulation models?**
   Process provenance and simulation are closely linked subjects. The execution of simulations, to forecast the outcome of a particular scenario or plan, necessitates a thorough record of all involved objects, processes and agents. Whereas provenance models are used to record such phenomena for later analysis, simulations models employ the same concepts to predict the behavior of the modeled systems in the future. Using PROV-O as a basis for SDSIM architecture inherently supports the creation of provenance data and can act as a common language for modeling provenance-like interactions of simulation systems[19].
7. **Can users attach business rules and similar restrictions to SDSIM architecture?**
   The SDSIM architecture Information Model provides the concept *Policy*, which represents a wide range of binding and non-binding statutes, contracts, business rules and guidelines. These can be connected to *Practices*, *Plans* and *Processes* that implement or follow these rules. Therefore business rules and similar restrictions can be incorporated into any automatic validation or simulation system. A possible way to express such rules in a semantic and machine-readable way is the Open Digital Rights Language vocabulary[20].

Perceptive analytics:

8. **Does SDSIM architecture support automatic recognition of events?**
   The automatic detection of events and patterns in real-time streams of data, a process also called streaming analytics, can provide real-time insights for decision logic and is the basis for many

[18] See www.w3.org/TR/shacl/.
[19] Suh, Young-Kyoon & Yong Lee, Ki. 2018. "A Survey of Simulation Provenance Systems: Modeling, capturing, querying, visualization, and advanced utilization." Human-centric Computing and Information Sciences. 8.10.186/s13673-018-0150-9.
[20] W3C. 2018. "ODRL Information Model 2.2." February 15. www.w3.org/TR/odrl-model/.

enterprise decisions. With the introduction of *Events* into SDSIM architecture, event-based analytics are supported natively. Multiple frameworks exist (such as FlinkCEP[21] or WSO2[22]) for detecting, filtering, and evaluating events, offering the possibility to instigate automatic responses and actions. Additional application could be derived from use in event-based process-mining strategies[23].

E. Application in digital business networks

1. **Can SDSIM architecture be exploited to achieve the digitalization of business-to-business transactional processes?**
   Providing a structured foundation for any digital message exchanged between partners in a supply chain is one of the distinct goals of this effort. The SDSIM architecture Information Model serves as a shared digital language used to define explicit bodies of structured, digital messages and establish an interface between partners. The SDSIM architecture Information Model allows users to unambiguously state the subject and object of the message in a technology- and industry-independent, as well as machine-readable, syntax. Automating the necessary exchange of messages between business partners or customers can reduce the time spent on the interaction of sales or customer relations operations significantly. Furthermore, the validation and automatic comparison between requirements for a given type of message and the actual received messages will further reduce the manual effort involved and extend current capabilities of supply chain management systems.
2. **Can SDSIM architecture provide high visibility between business partners, providing shared data of process, performance, inventory and product catalogs?**
   This competency question encompasses the complete scope of the SDSIM architecture model. It also represents the requirements for the fifth component of the Demand Driven Material Requirements Planning (DDMRP) approach, which recommends highly visible and collaborative execution of supply chain processes. The DDMRP approach is defined by Carol Ptak and Chad Smith in the third edition of "Orlicky's Materials Requirements Planning."[9] To achieve a highly visible execution of supply chain processes over organization borders, a simple exchange of messages in a predefined format is not enough. To accommodate semantic requirements for different industries, organizations, product domains, inventories, units and more, the SDSIM architecture Information Model must be extended dynamically and allow for fallback procedures. For example, a company may be in contact with suppliers and customers in different industries and using different SDSIM architecture extensions. In such a case, data interfaces have to adapt automatically to the required message formats and transform the available data into the required form. They have to react to changes in such requirements automatically. A solution is the definition of industry-, organization- or use-case-wide SDSIM architecture extensions and their automatic composition and exchange, based on semantic content negotiation, providing the same information in different formats. This approach is based on the HyperText Transfer Protocol content-negotiation mechanism that is used for serving different syntactic formats of web resources so that a user can specify which is best suited for the purpose.

3. **Can SDSIM architecture improve proactive risk management for a supply chain?**
   The enhanced visibility provided by an SDSIM architecture-centric business network and the addition of *Events* and their relations to *Processes* and *Metrics* will enable a much more analytical approach to risk

[21] The Apache Software Foundation. n.d. "FlinkCEP - Complex event processing for Flink." Accessed 2019. https://ci.apache.org/projects/flink/flink-docs-stable/dev/libs/cep.html.
[22] WSO2. n.d. "Stream Processor: An open source, cloud-native streaming data integration and analytics product optimized for agile digital businesses." Accessed 2019. https://wso2.com/analytics-and-stream-processing/.
[23] van Zelst, Sebastiaan J., Boudewijn F. van Dongen and Wil M. P. van der Aalst. 2018. "Event Stream-based Process Discover Using Abstract Representations." *Knowledge and Information Systems* 54, no. 2 (February): 407-435. https://doi.org/10.1007/s10115-017-1060-2.

management. The automatic evaluation of *Events* and the comparison with planned and actual states offers the reactive mode of operation a broad support by SDSIM architecture. If this is combined with analytical methods, combining predictions based on elapsed events, a proactive approach becomes possible.

2 The SCOR Digital Standard Information Model Architecture

This information model reflects the semantics gleaned from SCOR version 12 and extends the SDSIM architecture model, inheriting all SDSIM architecture semantics. At its core, this schematic formalization of the supply chain domain portrays the interplay among concepts of the four base taxonomies of SCOR, namely Performance, Processes, Practices and People. With the exception of Processes, all taxonomies are implemented as instance hierarchies[24], which means that all concepts defined in SCOR version 12 correspond with instances of the appropriate class, namely *Practice*, *Skill* or *Metric*. The instances of a *Process* are actual processes within an organization, unfolding at a specific place and time. Therefore, processes were modeled as a class hierarchy, representing such activities generically. In addition to these taxonomies, additional concepts, formerly only implied by the official specification, were introduced as generic delegates. They are necessary to close semantic gaps between concepts or to provide an important interface to related domains. (See section 2.6.)

Note: Under the scope of this initial conceptualization of SCOR, only level-1 and level-2 processes were included. In addition, the exact specifications of metrics in level 3 of the SCOR metrics taxonomy are not finalized according to the extended view on metrics. The first complete version of the SCOR Information Model, which will be released at a later date, will contain all of these specifics.

[24] Each of these taxonomies is implemented by employing the well-established SKOS vocabulary, which is directly incorporated into the SCOR Information Model, to document taxonomy internal relations. (See the table in the Namespaces section of the introduction for a link to the SKOS vocabulary.)

2.1 Practices

The practice section of the SDSIM architecture model is adopted without change into SCOR. In essence, the *PracticeType* concept was populated with a list of practice types, namely best practices, emerging practices, standard practices, obsolete practices and custom practices. They are represented by the following entities and descriptions:

- *BestPractices* are current, structured and repeatable practices that have had a proven and positive impact on supply chain performance.
- *EmergingPractices* introduce new technologies, knowledge or radically different ways of organizing processes. Emerging practices may yield a step change in performance by redefining the playing field within an industry.
- *StandardPractices* are how a wide range of companies have historically done business by default or happenstance. These well-established practices do the job but don't provide a significant cost or competitive advantage over other practices.
- *ObsoletePractices* are practices that were featured in a previous version of SCOR but are no longer considered a viable practice. These are included in this model as a means to keep track of old practices and also retire other practices in the future.
- *CustomPractice* is the concept a company can use to add organization- or use-case-specific practices that are not included in the SCOR model.

Note: A single practice can change its association to one of these types. Furthermore, this enumeration is not considered as complete or final and might change over time.

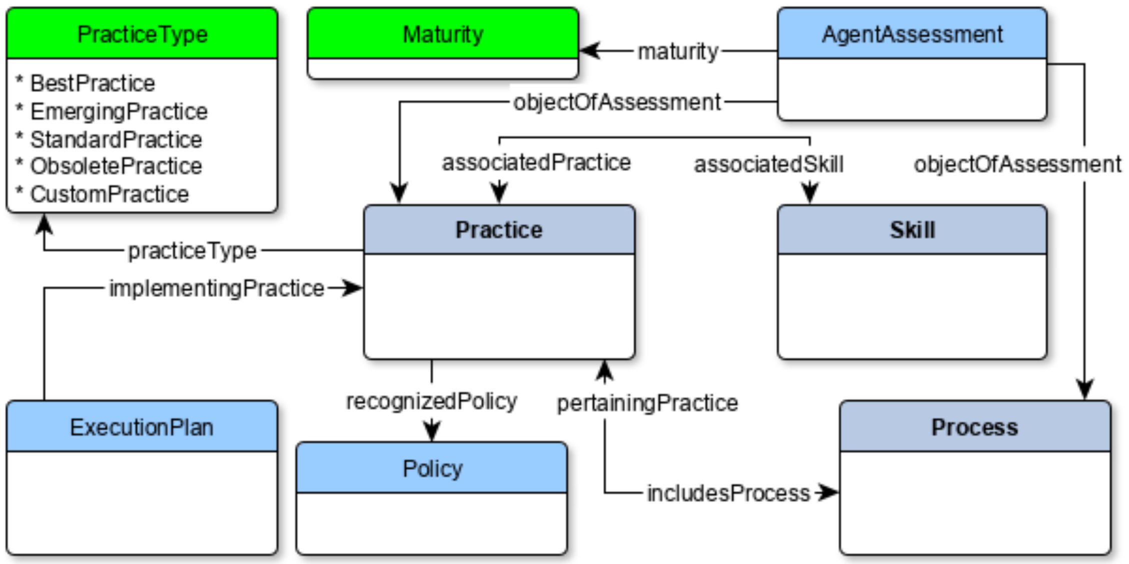


Figure 17: The SCOR Practices section

2.2 Skills

The Skills section of the SDSIM architecture conceptualization has been adopted without change into the SCOR Information Model. The flat Skills taxonomy of SCOR, which includes a list with no hierarchy, is supported by a list of recognized *Trainings* and known *Experiences* as available qualifiers for the *SkillProficiency* of an *Agent*.

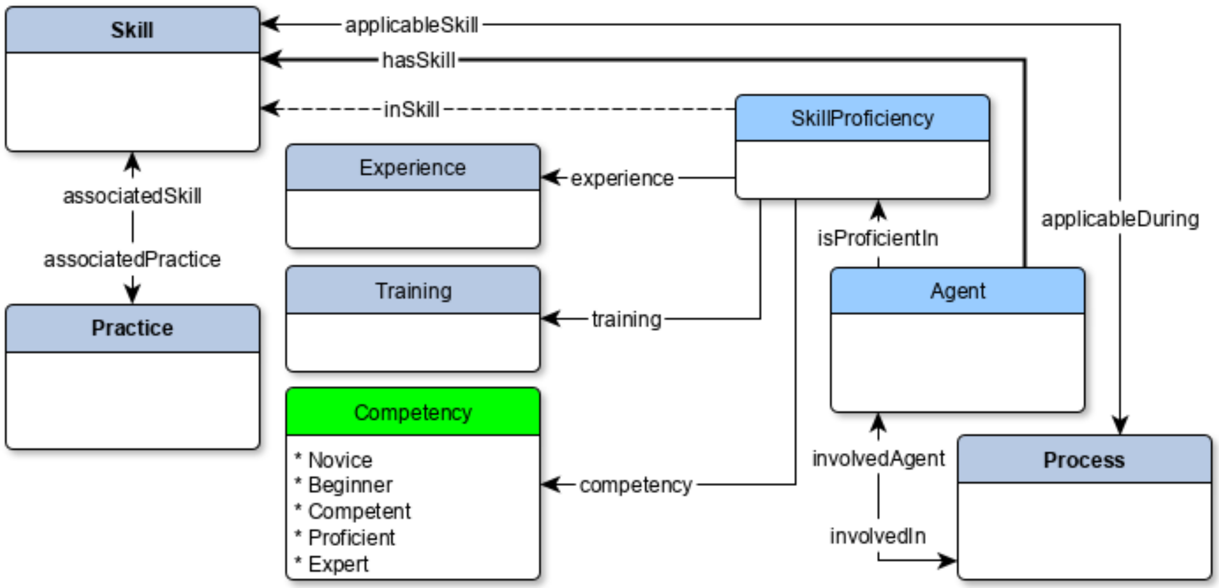


Figure 18: The SCOR Skills section

As with *PracticeType*, SCOR specifies five members for the Competency enumeration:

- A *Novice* is an untrained beginner with no experience who requires and follows detailed documentation to be able to perform the work.
- A *Beginner* performs the work with limited situational perception.
- A *Competent* individual understands the work and can determine priorities to reach goals.
- A *Proficient* individual oversees all aspects of the work and can prioritize based on situational aspects.
- An *Expert* has intuitive understanding and can apply experience patterns to new situations.

2.3 Metrics

The Metrics section of SDSIM architecture was reused without change in SCOR, although Figure 19 does not depict all of the related concepts from SDSIM architecture. SCOR details a host of *Metrics*, grouped by different *PerformanceAttributes*. *PerformanceObservations* can be gathered or calculated, in the case of *ComplexMetrics*, for each *Process* regarding a single *Metric*. In addition, an interface with the *Event* class was introduced to point out *EventCategories* that could have an impact on a given *Metric*. A detailed account of changes to the Metrics system in SCOR is available in section 1.6.

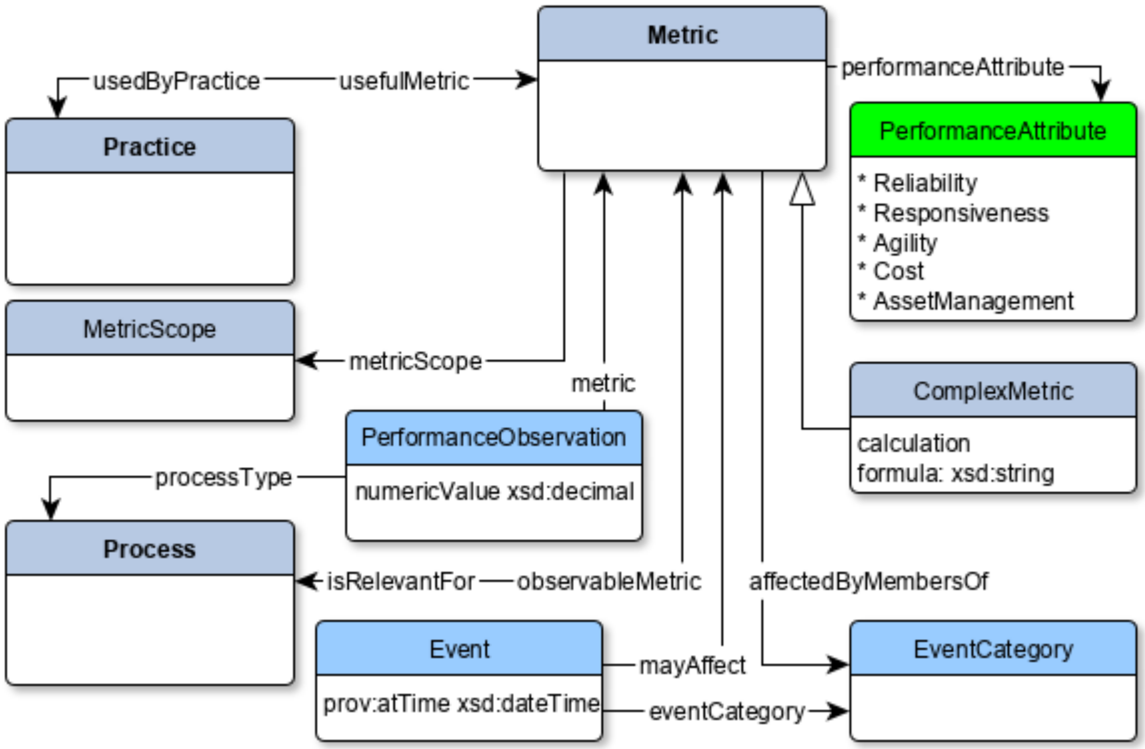


Figure 19: The SCOR Metrics section

Five *PerformanceAttributes* are defined within SCOR:

- The *Reliability* attribute addresses the ability to perform tasks as required. Reliability focuses on the predictability of the outcome of a process.
- The *Responsiveness* attribute describes the speed at which tasks are performed. Responsiveness addresses the repeated speed of doing business.
- The *Agility* attribute describes the ability to respond to external influences as well as the capability and speed of change.
- The *Cost* attribute describes the cost of operating the supply chain process.
- The *Asset Management Efficiency* attribute describes the ability to efficiently utilize assets. Asset management strategies in supply chain include inventory reduction and insource versus outsource.

2.4 Processes

The process hierarchy defined by SCOR is at the center of the business process reference model. It offers a standardized description of supply chain planning and operational processes at three levels of detail. SCOR models ***Processes*** as an elaborate class hierarchy closely aligned with its specification. Three additional classes — *Execute, SourceReturn and DeliverReturn* — were incorporated to lend additional structure to this taxonomy.

Execute

This class abstracts all execution *Processes* (*Source*, *Make*, *Deliver* and *Return*) under a single superclass of the process taxonomy. In its essence, this taxonomy can be summarized with *Plan*, *Execute* and *Enable*.

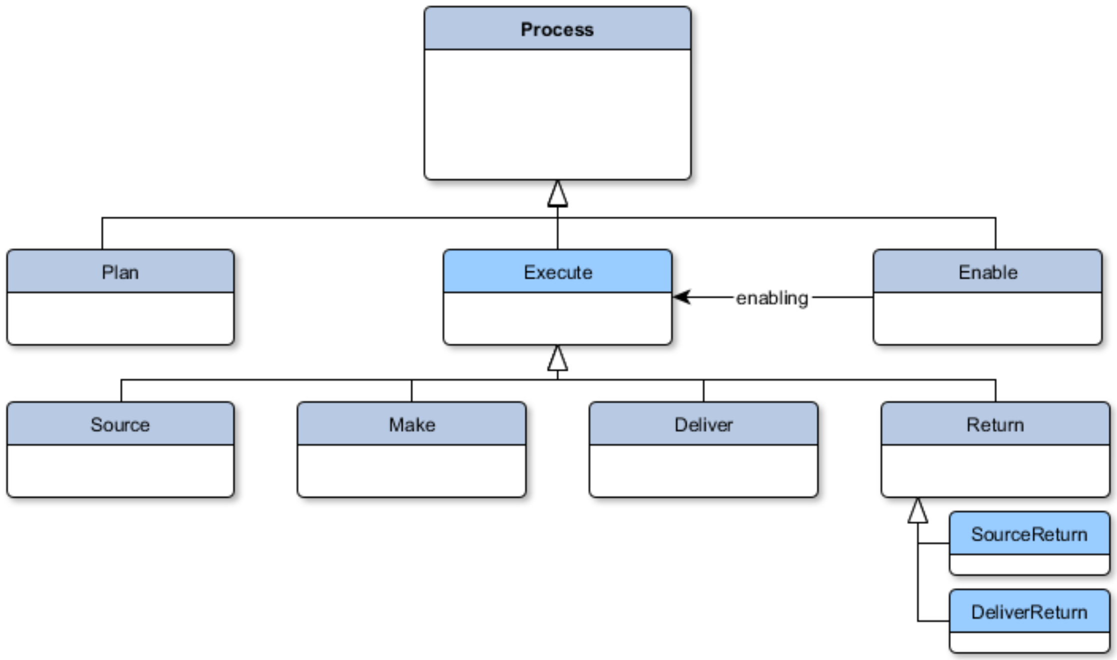


Figure 20: The top-level SCOR Process Hierarchy

SourceReturn and DeliverReturn

Both classes have only a structural function, similar to *Execute*, in that they divide the six defined subclasses of Return into two groups: upstream supply chain processes and downstream ones.

The SCOR model, including the first two levels of the process hierarchy, is depicted by Figure 22. The property *enabling* was introduced to relate an *Enable Process* directly to an *Execute* activity that benefited from or was otherwise influenced by this enablement.

2.5 ExecutionPlans

Note: *ExecutionPlan* is not a subconcept of the SCOR process type Plan. (See section 1.5.) It represents the result of such a planning process, often available in printed or digital form.

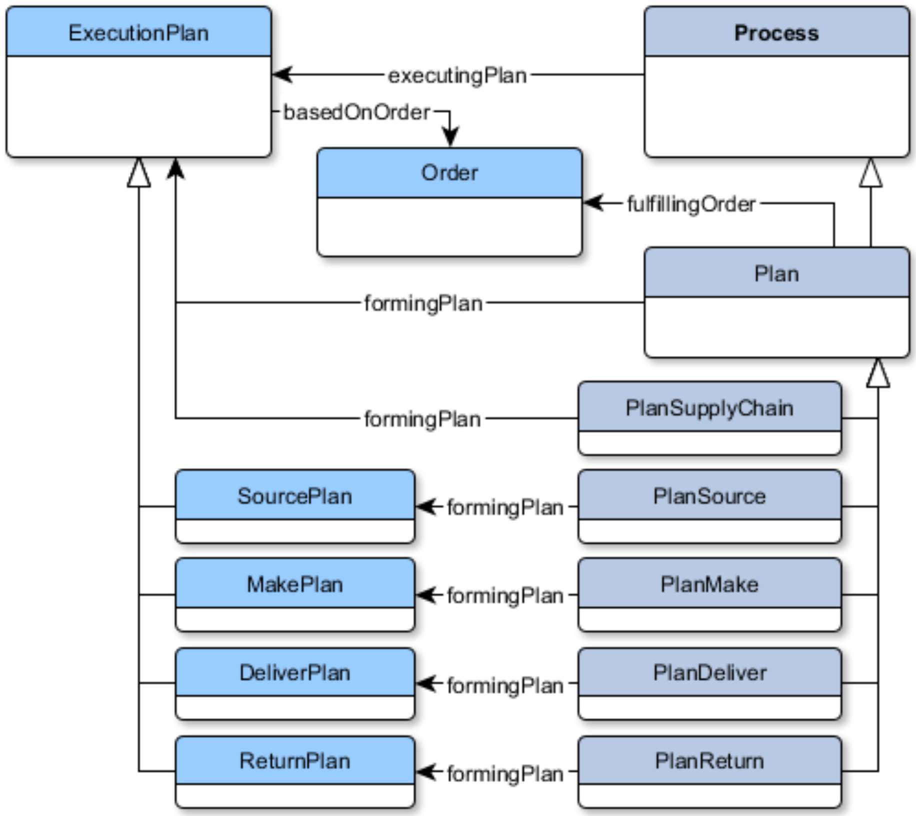


Figure 21: The SCOR Plans section

SCOR uses the SDSIM architecture concept *ExecutionPlan* to describe the results of *Plan Processes* by using the property *formingPlan*. An instance of *ExecutionPlan* may act as an additional input to any SCOR *Process* as a reference to a plan specification that was observed in the course of its execution, which is denoted by the property *executingPlan*. Parallel to the modular nature of *Processes*, *ExecutionPlans* can be decomposed into subplans. For that reason, four subclasses of *ExecutionPlan* were created to accommodate the existing subprocesses of *Plan* with specialized *ExecutionPlans* accordingly, namely *SourcePlan*, *MakePlan*, *DeliverPlan* and *ReturnPlan*. The property *basedOnOrder* expresses a dependency of an *ExecutionPlan* on a received *Order.* (See section 2.6.)

2.6 The Product and Order Interface

This part of the SDSIM offers a point of intersection with commonly used concepts of the production domain. As described previously, providing such high-level concepts without any further restrictions makes it easy to import external ontologies to fill these semantic gaps.

Order

An *Order* may refer to a wide range of orders, commissions or other types of instructions. In the context of supply chains, typical orders may include a purchase order, production order, sales order, transportation order, customer order, work order or other type of order. *Orders* usually consist of multiple *OrderItems*, which are denoted by *orderItem*, detailing a number of sub-items or tasks that make up the entire *Order*. A *Process* may indicate that it is executed to satisfy a given *Order* by using the property *fulfillingOrder*.

OrderItem

A single position of an *Order*. In the production context, this concept usually is used as a mediator, stating the amount or quantity; the type of product; and other optional qualifiers, such as quality specifications, specific production recipes to use or exact transportation requirements. A *Process* can point out which *OrderItem* is processed by the current execution by using the property *processedItem*. The property *product* refers to the product specification of this item.

Product

This concept is a generic class for product specifications. There are multiple efforts for standardized digital product specifications[25], including multiple ontology approaches[26]. The shape of the specifications is highly dependent on domain, participating organizations or product family. To accommodate this situation, no further restrictions are defined for this class. All relevant semantics can be provided with an external product specification vocabulary for that purpose.

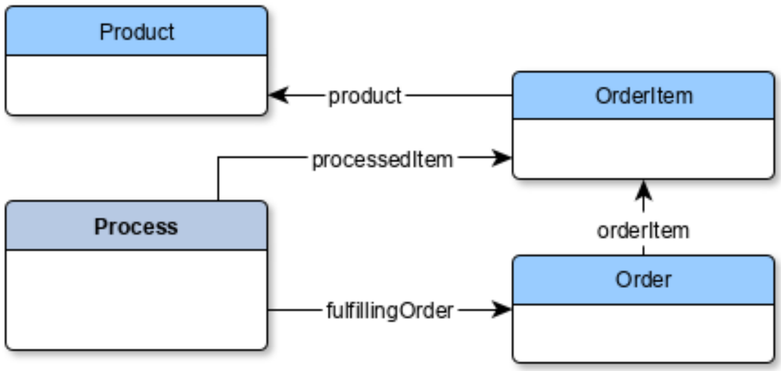


Figure 22: The SCOR Product and Order Interface

[25] Such as IEC 61360, which can be found here: cdd.iec.ch/cdd/iec61360/iec61360.nsf, and ISO 13584-42, which can be found here www.iso.org/standard/43423.html .
[26] An extensive list of known ontology-based specifications, including Central Product Classification, Classification of Products by Activity, United Nations Standard Products and Services Code, and the eCl@ss standard, can be found at www.ebusiness-unibw.org/ontologies/pcs2owl/.

This optional interface to a production environment can be used to link supply chain data to existing data sources, such as computer-aided manufacturing systems, enterprise resources planning systems, manufacturing execution systems and other production management solutions. The universal aim is to extend the graph-based view of product- and production-related data as far as possible to gain a comprehensive view of all elements of the domain, making them explorable, explicable and validatable. Many use cases depend on such universal visibility to detect possible disruptions of a supply chain early, pinpointing causes directly and guiding engineers to their root causes.

The SCOR Digital Standard Information Model in its current state is depicted in Figure 23. A finalized version of the SDSIM will be available in the near future.

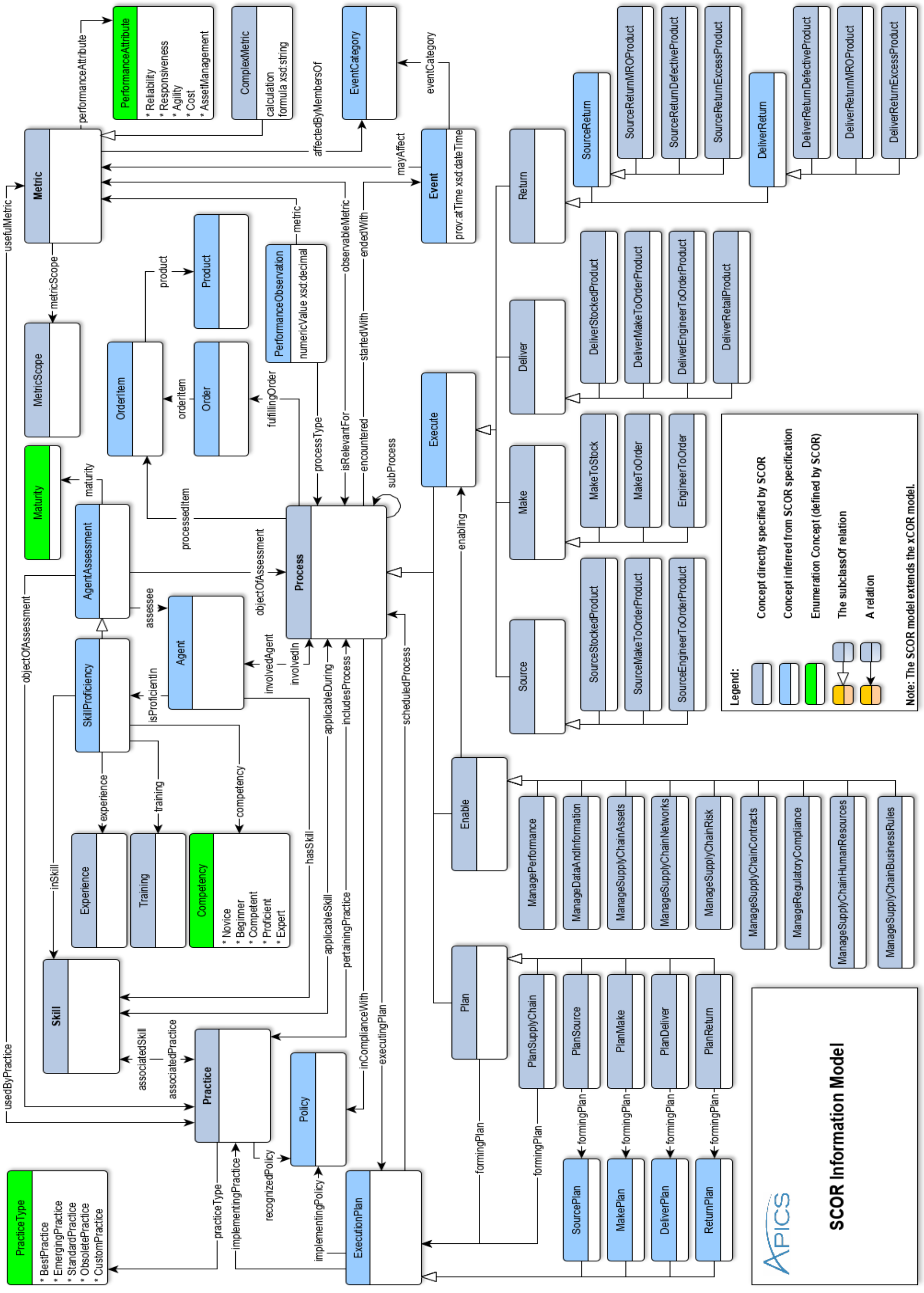


Figure 23: The SCOR Information Model