Date: January 2012

Date-Time Vocabulary (DTV)

FTF - Beta 1

OMG Document Number: dtc/2012-01-02
Standard document URL: http://www.omg.org/spec/DTV/1.0/PDF
Associated Schema Files: http://www.omg.org/spec/DTV/20111209
http://www.omg.org/spec/DTV/20111209/dtv-sbvr.xml
http://www.omg.org/spec/DTV/20111209/dtv-uml.xml
http://www.omg.org/spec/DTV/20111209/dtv.ocl
http://www.omg.org/spec/DTV/20111209/dtv.clif
http://www.omg.org/spec/DTV/20111209/sbvr.owl
http://www.omg.org/spec/DTV/20111209/sequences.owl

This OMG document replaces the submission document (bmi/2011-08-01, alpha). It is an OMG Adopted Beta specification and is currently in the finalization phase. Comments on the content of this document are welcome, and should be directed to issues@omg.org by June 4, 2012.

You may view the pending issues for this specification from the OMG revision issues web page http://www.omg.org/issues/.

The FTF Recommendation and Report for this specification will be published on September 21, 2012. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

DISCLAIMER OF WARRANTY

RESTRICTED RIGHTS LEGEND

TRADEMARKS

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials. Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but

may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page *http://www.omg.org*, under Documents, Report a Bug/Issue (http://www.omg.org/technology/agreement.htm).

Table of Contents

List of Figures

List of Tables

Preface

About the Object Management Group

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at *http://www.omg.org/*.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. A catalog of all OMG Specifications is available from the OMG website at:

http://www.omg.org/technology/documents/spec_catalog.htm

Specifications within the Catalog are organized by the following categories:

OMG Modeling Specifications

- UML
- MOF
- XMI
- CWM
- Profile specifications.

OMG Middleware Specifications

- CORBA/IIOP
- IDL/Language Mappings
- Specialized CORBA specifications
- CORBA Component Model (CCM).

Platform Specific Model and Interface Specifications

- CORBAservices
- CORBAfacilities

- OMG Domain specifications
- OMG Embedded Intelligence specifications
- OMG Security specifications.

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. (as of January 16, 2006) at:

OMG Headquarters
140 Kendrick Street
Building A, Suite 300
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*

Certain OMG specifications are also available as ISO standards. Please consult *http://www.iso.org*

Issues

The reader is encouraged to report any technical or editing issues/problems with this specification to *http://www.omg.org/technology/agreement.htm*.

1 Scope

Many SBVR rules involve common, generic, cross-domain concepts such as date and time. Characteristics of these concepts are frequent usage in everyday and business activities and wide usage across all business domains such as finance and manufacturing. These concepts exclude specialized needs such as sidereal time and real-time processing requirements. This document uses the term "foundation vocabularies" due to the foundational nature of these vocabularies for all these potential SBVR users.

SBVR tool vendors and users need standard vocabularies for such concepts to improve interoperability among vendors and to ensure that they share the same concepts in the same ways. Vendors also need an agreed format for exchange of date and time literals when used in rules. The SBVR community in general needs such vocabularies as a foundation to avoid the startup cost of defining vocabularies for basic concepts, and as an example for interoperability testing among tools. The OMG wants SBVR to be successful, and sees value in lowering the "cost of entry" for potential SBVR users.

This document addresses two different, but complementary, aspects of time:

Type 1: *Temporal noun concepts* (such as time coordinate, duration, calendar, etc.) that model attributes of SBVR noun concepts, and *temporal verb concepts* (such as time coordinate *is in the past*, time interval$_1$ *is before* time interval$_2$, time interval$_1$ *includes* time interval$_2$, etc.) that model relationships between temporal noun concepts. See clauses 8 through 8.2.

Type 2: Fact types that relate situation models and occurrences (such as a person being married to another person) to temporal concepts (e.g., to a time interval). See normative clause 8.3, as well as informative clauses 7.9 and 7.11, and informative Annex E.

These two aspects reflect the use/mention distinction well known from analytical philosophy: the first mentions temporal concepts, whereas the second uses temporal concepts in order to anchor situation models and occurrences in time.

The OMG's Model Driven Architecture (MDA) anticipates mappings between business-layer or Computation Independent Models (CIM) and implementation-layer Platform Independent (PIM) and Platform Specific (PSM) Models. To encourage such mappings, this document provides date and time models in UML (Unified Markup Language) plus OCL (Object Constraint Language), partially in CLIF (Common Logic Interchange Format), and partially in OWL (Web Ontology Language) modeled in ODM (Ontology Definition Metamodel). The UML, CLIF, and OWL/ODM date and time models are "equivalent" to the SBVR date and time vocabulary while being "true" to the spirit of their respective technologies.

2 Conformance

Conformance to this specification is defined with respect to three types of software:

1. Software that manages ontologies complies with this specification if and only if it can import the entire set of concepts defined by the Date-Time Vocabulary in at least one of the normative forms specified here.
2. Software that implements machine reasoning about time complies with this specification if and only if it interprets the entire set of concepts defined by the Date-Time Vocabulary according to the semantics defined here.
3. The compliance of software that interchanges documents containing date and time concepts is specified in Clause 13.

3 Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

- Bureau International des Poids et Mesures (BIPM), *The International System of Units*, 8th edition, 2006.
- International Electrotechnical Commission (IEC) 60050-111, *Physics and Chemistry, Edition* 2.0, 1996-07
- International Standards Organization (ISO) 8601, *Data elements and interchange formats - Information interchange - Representation of Dates and Times*, Third edition. December 1, 2004.
- International Standards Organization/International Electrotechnical Commission (ISO/IEC), JCGM 200: 2008, *International Vocabulary for Metrology - Basic and General Concepts and Associated Terms* (VIM), 3rd edition
- International Standards Organization (ISO), ISO/IEC 24707, *Information Technology - Common Logic (CL): a framework for a family of logic-based languages*, first edition, 2007-10-01
- International Standards Organization (ISO), ISO/IEC 80000-3, *Quantities and units -- Part 3: Space and time*, 2006
- International Standards Organization (ISO) 18026. *Information technology - Spatial Reference Model (SRM)*, 2009
- Object Management Group (OMG), *Object Constraint Language*, version 2.0, May 2006
- Object Management Group (OMG), *Ontology Definition Metamodel*, version 1.0, May 2009
- Object Management Group (OMG), *Semantics of Business Vocabulary and Business Rules* (SBVR), v1.0, January 2008, OMG document formal/2008-01-02.
- Object Management Group (OMG), *Unified Modeling Language* (UML), v2.3, May 2010
- World Wide Web Consortium (W3C), *OWL 2 Web Ontology Language Document Overview*, 27 October 2009
- World Wide Web Consortium (W3C) Recommendation, *XML Schema Part 2: Datatypes Second Edition*, 28 October 2004

4 Terms and Definitions

For the purposes of this specification, the terms and definitions given in the normative reference apply.

5 Symbols

This clause specifies the intended meaning of the symbols and other special text of this specification.

5.1 SBVR Structured English

This document adopts the “SBVR Structured English” syntax and font styles described in Annex C of the SBVR specification [SBVR]:

- Underlined teal indicates noun concepts.
- *Italic blue* identifies the fact symbols of verb concepts.
- Orange font indicates keywords.
- Double underlined teal marks individual concepts.
- Black normal font is regular text.

This specification uses the following symbols for the meanings indicated:

≤	less than or equal
≥	greater than or equal
<	less
>	greater
=	equal
+	addition
-	subtraction
*	multiplication
/	division

Ordinary arithmetic is meant when these symbols are used, unstyled, with numbers (e.g., "$number_1$ = $number_2$." The meaning is explicitly defined in this specification when these symbols are applied (and styled as verb concepts) to other operand types.

Sets are formed using the BNF syntax '{' <element>$^+$ (, <element>)* '}', where <element> gives the members of the set, separated by commas. An empty set is specified by "{}".

This specification uses the SBVR definition of '$thing_1$ *is* $thing_2$,' meaning "The $thing_1$ and the $thing_2$ are the same thing." Verb concepts using the fact symbol '*equals*,' '=,' or '*is equivalent to*' are explicitly defined for usages where the intended meaning is that two values can be distinct things, but are equivalent in terms of their relationship to some other thing. In particular, two quantity values are different things if they involve different units but are *equal* or *equivalent* if they *quantify* the same quantity.

The SBVR specification does not discuss dates and times, and thus does not specify the styling of literal time coordinates (e.g., "January 21 2009"), literal times of day (e.g., "3:00 pm"), and literal duration values (e.g., 3 months 13 days). These values identify themselves, meaning that each such expression identifies exactly one time coordinate, time of day, or duration value – they are what SBVR calls 'individual concepts.' For this reason, literal time coordinates and times of day are styled as individual concepts in this document. For example, January 21 2009 3:00 pm.

In this specification, duration values provide the reference scheme for durations, and time coordinates provide the reference scheme for time points. Verb concept roles that apply to durations or time points can be filled by duration values or time coordinates, respectively. For example, "17:00 *is* 1 hour *before* the start *of* the meeting" applies the verb concept "$time\ interval_2$ *is* duration *before* $time\ interval_1$" using time coordinate "17:00" to fill the "$time\ interval_2$" role, and duration value "1 hour" to fill the "duration" role. The example assumes that "start *of* meeting" is a time interval that fills the "$time\ interval_1$" role.

This specification distinguishes between comparing durations or time periods, and quantifying time periods. Comparisons uses verb concepts defined in this document and styled as verb concepts. For example, "if the length of the meeting *is greater*

than 3 hours …" or "if the date of the meeting *is before* the contract due date …" Quantifications use keyword style, as in "The party is on each July 4."

Definitions that are drawn from another specification are preceded by "Source" or "Dictionary Basis" captions. "Source" indicates that the definition is adopted exactly from the indicated specification. "Dictionary Basis" identifies definitions that are paraphrased from the specified source.

5.2 UML and OCL

This specification includes a matching and normative UML model that is inventoried in Annex A. The intent of the model is two-fold: (a) to illustrate this vocabulary with UML diagrams; (b) to satisfy the RFP requirement for a matching UML model.

The UML model is derived manually from the SBVR-based text in this document. In case of any discrepancies between the SBVR-based text in this document and the UML model, the text prevails because it is the original model.

The UML model is constructed generally following the principles in [SBVR] Clause 13. Names in the UML model match the corresponding SBVR names. UML name-quoting syntax is applied as necessary to quote names with embedded spaces. For example the SBVR term 'consecutive sequence' is quoted in UML as "consecutive sequence."

- Each SBVR vocabulary maps to a UML package.
- Each SBVR object type maps to a UML class.
- Each SBVR individual concept maps to a UML instance of the appropriate class.
- Each SBVR characteristic maps to a UML property of type Boolean.
- Each SBVR verb concept that uses the fact symbol *has* maps to a UML property.
- Other binary verb concepts map to UML associations and also to UML operations on one of the classes. The associations correspond to what SBVR calls the "noun form" of the verb concepts: an OCL expression can navigate across the association. The operations test whether two objects participate in the association and return a boolean result, supporting what SBVR calls the "sentential form" of the verb concepts.
- Verb concepts with more than two roles map to UML classes stereotyped as `<<fact type>>`. These are modeled with UML associations from the `<<fact type>>` to the UML classes that model the corresponding SBVR noun concepts, and also to a UML operation on the associated classes.

  The associations have cardinality '1' at the noun concept end because each fact type role is filled with exactly one instance for each instance of its fact type. The associations are one-way navigable from the fact type to the noun concept because normally one cannot navigate from an instance of a noun to all the fact types that involve the noun.

  The UML operations enable OCL expressions to directly exploit these concepts as functions.
- SBVR roles map to UML property names, association end names (rolenames), or to UML classes stereotyped with `<<fact type role>>`.
- SBVR Necessity rules that define cardinalities are modeled as cardinalities on UML properties or association ends.
- Where an SBVR object type is defined extensionally, the subtypes are shown using as a generalization tree to indicate that the subtypes are alternatives.

Several stereotypes are employed to relate UML model elements to SBVR concepts:

Table 5.1 - UML Stereotypes

Stereotype	Meaning
<<concept type>>	Labels a class to indicate that it models an SBVR Concept Type.
<<fact type>>	Labels a class to indicate that it models an SBVR Fact Type (Verb Concept).
<<fact type role>>	Labels a class to indicate that it models an SBVR Fact Type Role.

OCL constraints are incorporated into the document text and the UML model as follows:

- Each fully-formal SBVR definition has an equivalent OCL definition or constraint, captioned as “OCL Definition:”. The constraint captures the distinguishing characteristics of the formal definition. For example, if the formal definition of an SBVR object type ‘*luxury car*’ is ‘*car* that *is gold*,’ the corresponding OCL constraint is given as:

  OCL Definition:
  ```
  context "luxury car"
  inv:self. "is gold"
  ```

- Each SBVR Necessity (that is not a cardinality constraint) has an equivalent OCL constraint, captioned as “OCL Constraint:”.
- Necessities and Possibilities that specify cardinalities are modeled as UML cardinalities, rather than OCL constraints.

OCL is provided for Clause 12 and Annex D. These parts of the specification require the most rigorous definition.

5.3 CLIF Axioms

This specification includes a file of matching and normative Common Logic Interchange Format (CLIF) axioms that is inventoried in Annex A. The axioms are provided to precisely specify the formal Definitions and Necessities of this specification in a form that is meaningful to logicians and that can be input (in the future) to software that automatically checks for consistency among the axioms. The CLIF axioms in this document have been suntactically checked using the Kojeware CLIF validation service that is available at http://www.kojeware.com/clif-file-validator. No automated quality analysis has yet been performed.

The CLIF axioms are derived manually from the SBVR-based text in this document. In case of any discrepancies between the SBVR-based text in this document and these axioms, the text prevails because it is the original model.

Names in the CLIF axioms are based directly on the corresponding SBVR names, using CLIF name-quoting as necessary to address embedded spaces. For example the SBVR term ‘consecutive sequence’ is quoted in CLIF as “consecutive sequence.”

The file of CLIF axioms is derived automatically from CLIF statements that are incorporated directly in the text of this specification as follows:

- Each fully-formal SBVR definition has an equivalent CLIF axiom, captioned as “CLIF Definition:”. The axiom defines how the corresponding concept is derived from some other concept. For example, if the formal definition of an SBVR object type ‘*luxury car*’ is ‘*car* that *is gold*,’ the corresponding CLIF axiom is given as shown below. Read this as “each car is a luxury car if and only if the car is gold.”

  CLIF Definition:
  ```
  (forall ((car car))
   (iff ("luxury car" car)
      ("is gold" car))
  ```

- Each SBVR Necessity has an equivalent CLIF axiom, captioned as “CLIF Axiom:”. The axiom expresses the same constraint as the SBVR Necessity.

Many SBVR Necessities specify cardinality constraints. Basic CLIF cannot express these constraints in the absence of functions that generate collections, give the cardinality of collections, and compare the values of integers. Therefore this specification assumes the following in order to express cardinality constraints in CLIF:

- For each SBVR verb concept, there is a corresponding CLIF predicate, and also *n-1* CLIF functions, where *n* is the number of roles of the verb concept. The predicate and all the functions have the name of the verb concept, quoted if necessary. The distinction among them is the number of terms they take and which terms they take. The predicate takes one term for each role of the verb concept, and returns true or false according to whether the verb concept is satisfied for the specific terms. Each function omits one role and produces a collection of instances that fulfill that role in relationship to the other terms of the function.

  For example, given an SBVR verb concept ‘driver *drives* car *to* city,’ the predicate (“`driver drives car to city`” `John "car 123" Paris`) is true or false according to whether John drives car 123 to Paris. The function (`"driver drives car to city" John Paris`) returns the collection of cars that John drives to Paris.

- A primitive `count` function that returns the cardinality of a collection. For example, (`count` (“`driver drives car to city" John Paris`)) produces the number of cars that John drives to Paris.

- CLIF defines the = predicate as testing whether two terms are equal. This specification uses primitive functions <, <=, >, >=, and + to mean the standard numeric relationships. For example `(< (count ("driver drives car to city" John Paris)) 2)` tests whether John drives fewer than two cars to Paris.

- This document also uses the `allDifferent` function as defined in [IKL Guide].

CLIF is provided for Clause 12 and Annex D. These parts of the specification require the most rigorous definition.

6 Additional Information

6.1 Changes to Adopted OMG Specifications

This specification does not require or request any change to any other OMG specification.

6.2 How to Read this Specification

This document serves different purposes for first-time readers versus implementers. First-time readers should start with informative clause 7, "Rationale", which offers introductory text, and describes the motivations behind the design of this vocabulary. These readers may wish to refer to the normative clauses (clause 8 through clause 13), as well as informative Annex D, for definitions, notes, examples, and diagrams that describe the Date-Time Vocabulary concepts. The other Annexes provide additional examples and supporting information that should also be useful to these readers.

Implementers of this vocabulary will focus on the normative clauses and Annex D and on the supporting machine-readable files. The specific aspects of interest will depend upon the intended conformance goal, as described in clause 2. Implementers should study the material in the normative clauses in detail. The supporting informative material will also provide some guidance.

6.3 About this Specification

The first 6 clauses include information that is applicable to most OMG specifications. The rest of the document includes the following key topics:

Clause 7 - Rationale (informative) - introduces this document and discusses some of the key technical choices made by this specification.

Clause 8 - Time Infrastructure (normative) - describes fundamental concepts about time intervals, durations, and the relationships between those and situation models. This clause also provides supporting concepts needed to enable logical reasoning about time.

Clause 9 - Organizing Time (normative) - Time is conventionally organized by calendars, in order to simplify references to and enable arithmetic over time intervals and durations. Many time schemes and calendars exist, due to a variety of business needs, and historical, cultural, and religious traditions. This clause provides a foundation for others to define any time keeping or calendar system. It also defines the calendars that are standardized in [ISO 8601].

Clause 10 - Indexical Time Concepts (normative) - Indexical time concepts use terms such as “in the past” and “now” to refer to time. These terms are defined here, despite their inherent ambiguity, because they are frequently used in everyday communication.

Clause 11 - Duration Values (normative) - Duration values are amounts of time stated as multiples of time units, for example “5 hours 30 minutes.” The model of duration values presented here accommodates the complexities introduced by the varying number of calendar days in each calendar month and calendar year.

Clause 12 - Time Coordinates and Time Scale Convcrsions (normative) - Locations in time are generally specified by time coordinates, such as “July 31.” This clause defines the valid time coordinates for the standard calendars, and provides for conversions among various time scales as needed for arithmetic among duration values and time coordinates.

Clause 13 - Interchange of Duration Values and Time Coordinates (normative) - defines how duration values and time coordinates should be exchanged between tools that implement this specification. The interchange format is based on the existing [XML Schema] and [ISO 8601] specifications.

Annexes

Annex A: Attachments (normative) - Lists the machine-readable files that accompany this specification.

Annex B: References (informative) - this section lists the standards documents and academic papers that were consulted in the preparation of this specification.

Annex C: EU-Rent Date-Time Example (informative) - gives several worked-out examples illustrating this specification.

Annex D: Fundamental Concepts (informative) - International standards, for example [VIM], [ISO 80000:3], and [ISO 18026] define duration as just one of many quantity kinds, and time scales as one of many kinds of coordinate systems. This permits the formation of derived quantites based on durations (e.g., velocity, which is length / duration), and multi-dimensional coordinate systems that include time as one dimension. Coordinate systems themselves depend upon mathematical concepts, such as sequences and scales. Unfortunately, there is no existing SBVR vocabulary or ODM ontology that addresses these concepts. The authors recognize that they are out-of-scope for this specification, but felt it necessary to imagine how this Date-Time Vocabulary would fit into a complete schema that addresses them. Annex D summarizes that schema in the form of several SBVR vocabularies.

This material is informative and is provided to illustrate how the authors think this Date-Time specification fits in a larger context. The team expects that some future effort may work on standardizing these concepts, at which time this specification should be revised to fit properly within the normative version of such concepts.

Annex E: Formalizing English Tense and Aspect (informative) - The normative clauses of this specification deal with the semantics of time as used in natural languages. This Annex describes how propositions that are given in English language syntax may be formulated using the Date-Time Vocabulary.

Annex F: Simplified Syntax for Logical Formulations (informative) - describes a simplified syntax for representing SBVR logical formulations.

Annex G: OWL/UML Diagrams (informative) - contains a number of UML diagrams that illustrate the attached OWL ontologies.

Annex H: Index of Business Designations (informative) - contains an index to the business designations defined in this document.

6.4 Acknowledgements

The following companies submitted and/or supported parts of this specification:

- Automata, Inc. - Paul Haley
- Business Rule Solutions, LLC - Ron Ross
- Business Semantics, Ltd - Donald Chapin
- Deere & Co - Roger Burkhart
- Hendryx & Associations - Stan Hendryx
- International Business Machines - Mark H. Linehan (team lead)
- KnowGravity Inc - Markus Schacher
- LogicBlox - Terry Halpin
- Microsoft - Don Baisley
- Model Driven Solutions - Cory Casanave
- Model Systems - John Hall
- National Institute of Standards and Technology - Ed Barkmeyer
- PNA Group - Sjir Nijssen
- Ravi Sharma
- Thematics Partners - Elisa Kendall

7 Rationale

This Informative clause introduces this document, and discusses various design considerations that impacted it.

7.1 Multiple Goals

This vocabulary attempts to satisfy several goals that tend to conflict.

1. Provide a Standard Business Vocabulary for Date and Time Concepts. Provide a vocabulary of date and time concepts that business users can share and exploit in their business domain vocabularies and rules. Quoting Donald Chapin, this requires an "... SBVR Foundation Business Terminology that is conceptualized optimally for the way people think and communicate about things in their organizations using natural language." To satisfy this goal, the date and time vocabulary needs to include terms that make intuitive sense to business users.

2. Support Machine Reasoning about Time. Provide a formal ontology that enables machine interpretation and reasoning. This means that processing by automated reasoners is possible, based on a well-grounded formal representation. For example, it should be possible for a reasoning system to determine whether a payment is more than 30 days late compared to some due date. Satisfying this goal requires carefully-defined vocabulary concepts, to the point of making distinctions that would not occur to business users. The business vocabulary is grounded on the formal ontology, so these distinctions show through in the business vocabulary.

3. Enable implementation. Enable tool vendors and other software developers to implement the date and time vocabulary with a "reasonable" amount of development effort – meaning that the value obtained is commensurate with the development cost. That cost is driven by the size of the vocabulary – the more there is to implement, the greater the cost. Implementation cost is also driven by the effort required to resolve ambiguities, omissions, and inconsistencies in the specification. Including a formal grounding and concise vocabulary is expected to facilitate both development of tools and use of the specification by vendors, business users, and those who want to apply formal reasoning systems.

This specification employs several techniques to reconcile these different modeling goals. The vocabulary is presented as an SBVR business vocabulary, with extensive examples and notes. Many formally-defined concepts are also presented in CLIF and OCL. Wherever possible, terms and examples are chosen to make sense to business users. Parallel construction of terms ensures that related terms are used consistently. Every concept is precisely defined. Multiple distinct concepts are defined where needed to distinguish between concepts that are intuitively similar but have different reasoning implications.

Annex D, "Foundational Concepts" documents general concepts that, though out-of-scope for a date and time vocabulary, nevertheless must be implemented consistently by reasoning systems. Annex D includes formal mathematical definitions of sequences, on which all scales, not just time scales, are based, and a general treatment of quantities and units, and of basic mereology. Although Annex D is not normative, it will provide guidance that should ease formal integration of future possible normative specifications, perhaps published by the OMG or other standards bodies, of the Annex D concepts with the normative vocabulary of this specification. Implementors of this specification are encouraged to support or assure compatibility with Annex D. Normative concepts of this specification that specialize Annex D concepts formally includes Annex D concepts in their definitions, as if Annex D were normative.

Implementors and reasoning systems are also addressed by providing this date and time vocabulary in SBVR, UML, and CLIF forms.

7.2 Reckoning of Time

The scientific community, and some time standards such as OWL-Time, typically conceive of time as continuous, meaning that any moment of the Time Axis can be subdivided into an infinite number of smaller moments. This Date and Time Vocabulary follows that pattern by modeling time as a segment of the Time Axis called a time interval, and describing amounts of time as durations.

Mathematically, both time intervals and durations correspond to contiguous sets of real numbers, making modeling of time-varying phenomena amenable to continuous mathematics. This specification gives a rigorous account of the operations that may be performed on time intervals and durations, providing the basis for formal reasoning about time.

Since antiquity, the passage of time has been reckoned by counting discrete time intervals demarcated by the diurnal and annual cycles of the Earth and the Moon's cycle – giving rise to 'time point' concepts such as 'calendar day, 'calendar month', and 'calendar year'. To identify a particular element of a cycle, each cycle is mapped onto a 'calendar'.

Calendars define time scales used refer to time points by name or by scale index. The combination of a time scale and an index or a name (e.g., 'February') is called a 'time coordinate'. An individual time coordinate is called an 'atomic time coordinate', whereas combinations of time coordinates (e.g, "February 3") are called 'compound time coordinates' (clauses 7.5 and 12.1.2). Time coordinates provide a reference scheme for time points via the verb concept 'time coordinate *indicates* time point'. Thus time points can be referred to either by definition descriptions (e.g, "the day after the meeting") or by time coordinates (e.g., "3:00 p.m.").

Each time point is a concept whose instances are time intervals. Thus, every 'time interval' fact type role in this specification can be filled by a time coordinate that *indicates* a time point. For example, the statement "the meeting time *is before* 3:00 p.m." uses the "time interval$_1$ *is before* time interval$_2$" verb concept (clause 8.1.2) to compare one time interval given as a definite description with another time interval given as a time coordinate.

Many calendars have been devised, ancient and modern. Time coordinates of most calendars can be correlated to jointly reference the same time interval. Calendars are anchored to the Time Axis by associating a noteworthy event with a particular time point on the calendar, e.g., the signing of the Convention du Mètre in Paris on May 20, 1875, which established the International Bureau of Weights and Measures (BIPM), and is the anchoring event for the modern Gregorian Calendar.

It is noted that, except for the second, none of the units of time are of exactly the same duration. This variation arises because the periods of rotation and revolution of the Earth and Moon are incommensurable and irregular, significantly complicating the task of timekeeping. These variations are accounted for by incorporating intercalary leap days and leap seconds into some calendar and timekeeping schemes. This specification chooses to support leap days, but not leap seconds, because leap days are significant to business while leap seconds are insignificant.

Time is measured by clocks, or tracked by calendars, in discrete time intervals called 'time periods', which *instantiate* time point sequences, as discussed in the next section. A particular member of a time scale – and a time period that *instantiates* a time point sequence of just one member – is called a 'time point'. Every time scale divides the Time Axis into time points with a specified duration, called the 'granularity' of the time scale. One consequence of this model is that every time period is aligned to the time points of a time scale: the time period *starts on* the first time point of some time point sequence of the time scale, and the time period *ends on* the last time point of some time point sequence of the time scale. Another consequence is that the duration of every time period is a multiple of the granularity of the time scale.

Of course, any time point can be subdivided by another time scale with a finer granularity. For example, a time point with duration "1 second" can be divided into milliseconds. But subdivision in this sense is still a discrete process. The finer time scale has a finite number of time points for each time point on the original time scale.

In everyday activity, people and businesses talk about durations such as years and hours, and about time periods such as calendar years, hours of day, and so forth. These discrete time concepts are used in ordinary conversation, in business

contracts, in legislation and regulations, and in corporate policies. They also form the basis for identifying time intervals for scientific purposes (International Atomic Time) and for navigation (Global Positioning System). Representation of time in computers is inherently discrete and finite. Consequently, this specification also defines discrete time modeled by time scales.

7.3 Time Scales

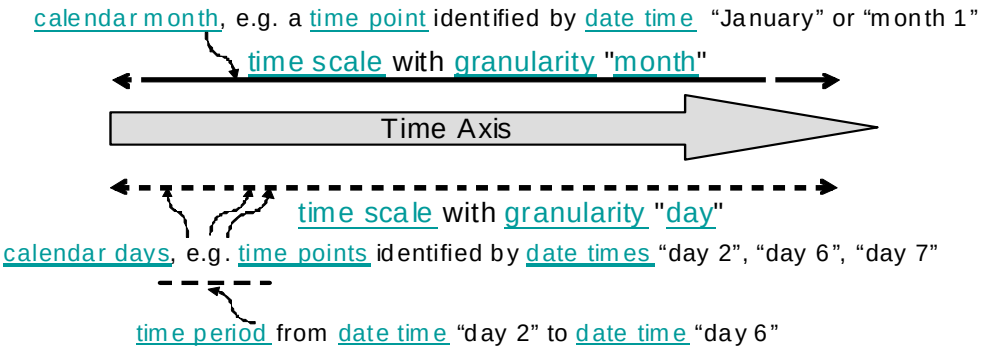

Figure 7.1 - The Time Axis and Time Scales

Following [ISO 8601], this specification considers that there is a single Time Axis that is measured by multiple time scales. The Time Axis represents "the succession in time of instantaneous events". Figure 7.1 shows the Time Axis with one time scale for calendar months, and another for calendar days.

Each time scale comprises a consecutive sequence *of* time points at regular or irregular time intervals. The time points of each time scale have a duration that is called the granularity of the time scale. Month scales have irregular time intervals because different calendar months have different durations. Thus, the Time Axis is continuous time, while time scales partition the Time Axis into discrete segments. Time scales define concepts that are meaningful in business and everyday life.

Time coordinates label individual time points on a time scale. For example, the top time scale in Figure 7.1 has a calendar month labeled "January", while "day 2", "day 6", and "day 7" are indicated on the time scale for calendar days. A time coordinate can have multiple labels. For example, "January" is also labeled "month 1".

A time period *instantiates* a time point sequence, a sequence of consecutive time points on a time scale. "Instantiation" means that the time point sequence *corresponds to* the time period, analogous to SBVR's "meaning *corresponds to* thing". Each time point sequence has a first time point, a last time point (the final time point of the time point sequence), and a duration (the length of the time period). For example, the time point sequence from "day 2" to "day 6" has a first time point of "day 2", a last time point of "day 6", and a duration of "5 days".

Conventionally, and by international agreement, on some time scales (hours, minutes) the first time point is designated "hour 0" or "minute 0", while on others (months, weeks, days) the first time point is designated "month 1", "week 1", or "day 1". Historically and in [XML Schema], calendar years are numbered from 1 but scientific practice and [ISO 8601] counts a year 0.

Conversion between time scales is possible via formulae that specify how a time point on a coarser time scale indicates the same time interval as a time period on a finer time scale.

7.4 Distinctions

The distinction among time coordinate and duration values is significant. A time coordinate gives a location on a time scale. A duration value specifies an amount of time. For example, a meeting might occur at "3:00 p.m." (a time coordinate) for "3 hours" (a duration value). This distinction leads to separate terms for concepts such as "day" (a time unit used with duration values) and "calendar day" (a time point indicated by a time coordinate).

There is a many-to-one relationship between time coordinates and time points. For example, "January 2009" and "month 1 of 2009" are two time coordinates for the same time point. In SBVR terms, time coordinates provide the reference scheme for time points. In human language, a thing and a reference to the thing are often not distinguished, but the difference is important in ontological reasoning.

Similarly, there is a many-to-one relationship between duration values and durations. "1 hour" and "60 minutes" are two duration values for the same duration. Again, the distinction is significant ontologically but often blurred in human discourse.

7.5 Compound Time Coordinates

Compound time coordinates are time coordinates composed from multiple time scales. Compound time coordinates are used to designate a time interval whose duration is much less than the span of a time scale. For example, to identify a particular calendar day on a time scale that spans millennia, the compound designation "3 January, 2010" is used, rather than something like "day 733 795". Compound time coordinates originated historically as counts of the apparent cycles of the Sun, the Moon, and the stars.

Around the globe, different cultures express compound time coordinates in different ways. For example, "January 3, 2010", "3 January 2010", "2010-01-03", "1/3/10", "3/1/10" represent the same date in different parts of the world. Similarly, the same time may be expressed as "6:00 p.m." or "18:00". For example purposes only, this document gives dates and times in various formats. However, this specification does NOT standardize any particular way of expressing dates and times. (See [ISO 8601] for such a standard.) Instead, this specification focuses on formally capturing the meaning of compound time coordinates that may be expressed in various date and time formats and in different languages.

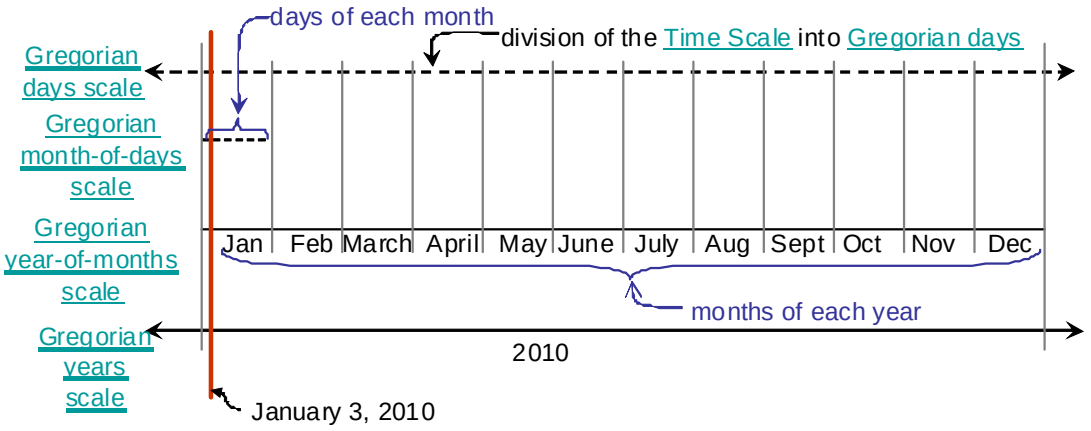

Figure 7.2 - Example of Gregorian calendar

This specification models "2010" as a time coordinate (a date time, or timestamp) on the Gregorian years scale. It models "3 January 2010" as a compound time coordinate that references multiple time scales of the Gregorian calendar. The compound time coordinate specifies time points on the Gregorian years scale, the Gregorian year-of-months scale, and the Gregorian month-of-days scale. Put together, these time points on these time scales *indicate* (by definition of 'compound time point') a particular time point on the Gregorian days scale.

Notionally, the Gregorian days scale is a time scale of granularity 'day' that extends indefinitely into the past and the future. "3 January 2010" can be understood as the time interval *indicated by* a particular time point on the Gregorian days scale. Clause 12.1 gives details about this. Clause 12.4 gives details about conversions between time scales.

Not all time scales can be combined in compound time coordinates. For example, "day 33 second 45" makes no sense. Clause 12 details both the time scale combinations that form legitimate compound time coordinates, and their meaning in terms of atomic time coordinates. For example, "01:35" is a compound time coordinate (using the day of hours scale and the hour of minutes scale) that means minute of day 95 on the day of minutes scale.

The meaning of some compound time coordinates as time intervals depends upon the presence or absence of leap days. For example, the relationship of March, April, etc. on the Gregorian year-of-months scale to the Gregorian days scale depends upon the number of days in February. In leap years, there is an additional day in February that "bumps" March over by one day on the Gregorian days scale. Hence, a compound time coordinate such as "3 March" does not mean a single Gregorian day on the Gregorian days scale if the calendar year is not given. Instead, such a date is understood as a choice among two possible Gregorian days. The choice is called a 'time set' and denoted (in this example) as "{Gregorian day 62, Gregorian day 63}".

7.6 Compound Duration Values

Compound duration values are duration values composed from multiple time units. Examples are "3 weeks 4 days", and "1 hour 30 minutes". The meaning of these is durations using the smallest time unit of the compound duration values. For example, "3 weeks 4 days" means "25 days", and "1 hour 30 minutes" means "90 minutes".

Some compound duration values that use nominal time units are ambiguous. For example, "5 months 3 days" is ambiguous because the number of Gregorian days in a Gregorian month of year varies. Similarly, the number of Gregorian days in a Gregorian year varies according to whether the Gregorian year is a leap year. The concept 'duration value set' models the ambiguity. For example, "2 years 1 day" means the duration value set {730 days, 731 days}.

7.7 Granularity of Time Coordinates and Time Points

The granularity of a time coordinate is understood as the finest granularity of the components of the time coordinate. For example, the granularity of "3 January 2010" is 'day'. This is important when understanding the meaning of a phrase such as "the meeting happens on 3 January 2010". The phrase means that the meeting happens sometime during that calendar day, but does not say whether it happened at noon or 18:00 or throughout the entire calendar day because the granularity means the whole day. A phrase such as "the meeting happens at 18:00 3 January 2010" is more specific because it uses a compound time coordinate with granularity 'hour'. It means that the meeting happens sometime within the hour indicated by "18:00". To specify the time more precisely, add minutes or seconds or even fractional seconds to the compound time coordinate to achieve the desired temporal resolution. The granularity chosen in giving a time coordinate should be as specific as required for any particular use case.

Similarly, the time unit of a compound duration value is the least time unit of the individual atomic duration values that makeup the whole duration value. For example, "6 hours 00 minutes" has a time unit of "minute", while "6 hours" has a time unit of "hour".

7.8 Time Point Relationships

This specification provides relationships among time points and durations that permits comparing, adding, and subtracting them in various combinations. These are described in Clause 12 in terms of fundamental relationships (e.g., the mereological aspects of time intervals, the *is before* relationship between time intervals, the Allen relations), and various derived relationships.

Some duration value relationships, when applied to operands that have nominal time units, may have no meaning. For example, it makes sense to compare two duration values that are in months with each other (e.g., "5 months *is greater than* 3 months"), but comparing some duration values in months to some duration values in days (e.g., "2 months is ? than 59 days") may be meaningless since months have varying numbers of days. Whether a relationship has meaning may depend upon both the values and time units of the relationship operands. For example, "10 days *is less than* 1 month" is always true, even though individual Gregorian months may be 28, 29, 30, or 31 Gregorian days. Clause 15 addresses these issues.

Similarly, time relationships may be ambiguous when applied to time coordinates or time points. For example, the time interval from 8 January through 13 March (given without the Gregorian year) has one of two durations, the duration value set {65 days, 66 days}. Clause 16 discusses these complexities.

7.9 Temporal Reasoning

A major goal of the Date-Time vocabulary is to enable reasoning about time in fact models. Such reasoning presupposes that the temporal aspects of each sentence are described in the logical formulation of the sentence. This section provides a summary of issues involved and describes how this specification supports temporal reasoning. A more thorough treatment is provided in clauses 8.3 and 10.3.

Fundamentally, time is associated with events and with the lifecycle of things. This specification uses the term "situation" to refer to events, activities, states, etc. Linguists often categorize situations in various ways, for example as "events", "situations", "actions", and so forth. This specification chooses not to categorize situations, but instead to focus on various relationships between situations and time.

Situations are said to *occur*, which is a primitive notion. Some situations that are conceptualized never occur. This specification uses the term 'occurrence' for a situation that occurs at some time in the world that is taken to be actual. When one is making a decision in the real world, what is taken to be actual is what the decision maker knows or believes about the real world. When one is analyzing a what-if situation (as in a business plan), the hypothetical elements of that situation are taken to be 'actual'.

When something occurs, there is always a time associated with the occurrence. The time may be present, past, or future, relative to the decisions being made. This permits distinctions among different instances of some situations that recur. For example, "Oceanic Air flight 815 flies from NY to Los Angeles" may be a situation that occurs many times and for which the individual occurrences may be distinguished by time. However, many types of occurrences are not distinguishable by time. For example, multiple child births often happen at the same time, so are not distinguishable purely by time.

The basic element of time introduced in Date-Time is a time interval, a portion of time having a non-zero duration. One basic fact type relates occurrence to time interval: 'occurrence *occurs throughout* time interval'. It represents the idea that the occurrence is ongoing at every point in the time interval. From it, we derive the characterizing relationship 'occurrence *occurs for* time interval' (clause 8.3.1). This fact type represents the idea that the occurrence starts at the beginning of the time interval and ends at the end of that time interval. For any occurrence, there is exactly one such time interval, called the occurrence interval.

A situation model is the conceptualization of a situation that could occur in some possible world. In a given world of interest (the world taken to be actual), each situation model has zero, one, or more occurrences. We say that an occurrence

exemplifies a situation model. The situation model itself is said to *occur for* each time interval that is the occurrence interval of an occurrence of the situation model. Other verbs that relate occurrences to time intervals are used to relate situation models to time intervals by extension. The critical difference is that an occurrence is a single actual situation and *occurs for* exactly one time interval; a situation model is an abstraction of zero or more occurrences and may *occur for* zero or more time intervals, one for each distinguished occurrence.

Occurrences are partially ordered by the times of their occurrence – their occurrence intervals. This specification provides the basic vocabulary to describe the ordering of occurrences in clause 12.3.4. Ordering of occurrences allows some statements to be made about the ordering of situation models, and those verbs are defined in clause 12.3.6.

This document uses 'proposition' as it is used in formal logic, to mean the expression of a complete thought that has a logical interpretation – a sentence. Each proposition (that is not paradoxical) *describes* exactly one situation model. This viewpoint was famously championed by Donald Davidson, that a proposition is a definite description of a situation ([Davidson], p. 504). This specification adopts this viewpoint. A proposition is either true or false in a given world. A situation model either *occurs* or does not *occur* in the world that is taken to be actual. There is a duality in that a proposition may simultaneously have a truth value and describe a situation model. Every proposition describes a situation, whether that situation exists or not. So we say that every proposition describes exactly one situation model. The proposition that describes a situation model that has more than one occurrence in the world taken to be actual, however, is sometimes true and sometimes false in that world. The truth value of such a proposition is an open question. This specification does not deal with the issue of assigning truth values to propositions; it is only concerned with propositions as descriptions of situations.

Since a proposition describes exactly one situation model, it is said to *describe* every occurrence of that situation model as well. In many cases, this is the critical fact type: proposition *describes* occurrence. For example, "the books of corporation XYZ are reviewed annually at corporate headquarters" can be formally represented as:

> In every fiscal year (a business-defined time period), there is an occurrence that is described by the proposition "the books of corporation XYZ are reviewed", and that occurrence occurs at the corporate headquarters.

A statement contains explicit and implicit references to time that restrict the time interval of the situation it describes. Time is inescapable in a temporal model, it is pervasive. There is a time interval(s) associated with every fact statement, explicitly or implicitly. Explicit references are time coordinates, indexicals, and definite descriptions. References to time are implicit in the tense and aspect of verbs. This specification includes definitions of time coordinates, indexicals, and calendar terms used in statements, and formulations for the most common tenses and aspects.

Each example given above assumes that the relevant concepts are defined in domain-specific vocabularies. Such vocabularies include verb concepts, such as "flight *takes off*". Human languages use many different prepositions ("at", "on", "in", "during", etc.) for relationships with time. This specification supports verb concepts with a few of these prepositions, with the expectation that business vocabularies will define verb concepts using other prepositions as appropriate for particular business domains.

7.10 Temporal Granularity

The granularity of a time point is important to the semantic meaning of a statement such as "Apollo 13 *launched on* 11 April 1970". Since we know from background knowledge that the launch took much less than a day, we understand this as "the occurrence 'Apollo 13 *launched*' happened within the specified calendar day". Public records show that Apollo 13 actually launched at "14:13 EST" on that day. But the statement "Apollo 13 *launched on* 11 April 1970" does not give any hours or minutes; it just gives the day. It tells us that the occurrence happened sometime during the day or perhaps throughout the day. It tells us no more. If given as "Apollo 13 *launched on* 11 April 1970 at 14:13 EST", and assuming the launch took less than a minute, then we would know the time with minute granularity, that is that the launch happened within the specified minute of hour.

7.11 Language Tense and Aspect

Most human languages incorporate *tenses*, to indicate whether propositions occur in the past, the present, or the future with respect to the time of utterance of the proposition. For example, "company x *traded with* company y" is past tense. This specification captures the semantic meaning of tenses by associating situation models and occurrences with time and then indicating whether that time is past, present or future with respect to current time. For example "company x *traded with* company y" is understood as "the occurrence 'company x *trades with* company y' *is in the past*". This approach to formalizing human sentences about tense follows [Parsons].

Many human languages also incorporate *simple*, *progressive*, and *perfect* aspects. *Simple aspect* applies to activities independent of whether they are ongoing or completed. For example "company x *traded with* company y", meaning that the two companies did trade, but does not say whether the trading is ongoing or completed. *Progressive aspect* means that an activity was ongoing or is ongoing or will be ongoing. For example "company x *was trading with* company y", meaning that the trading was continuing.

Perfect aspect indicates that an activity is accomplished. For example, "company x *will have traded with* company y" says that at some time in the future, the trading activity will be achieved. The difference between the simple and perfect aspects is shown by comparing the phrases "John *writes* a book" and "John *has written* a book". The second example, using "*has written*" applies the perfect aspect to indicate that the writing is complete. The first example, using "*writes*" uses the simple aspect. It does not say whether the writing is finished.

The progressive and perfect aspects may be combined to indicate that an activity both was ongoing, and is achieved. For example, "John *has been writing* a book" indicates that the writing occurred over time and the writing is completed or achieved.

In this specification, the progressive and perfect aspects are formally captured by characteristics of situation models and occurrences: "situation model *is continuing*" and "situation model *is accomplished*". Thus, any situation model may be progressive or not, and may be perfected or not. Both are independent of whether the situation model is in the past, the present, or in the future.

Human languages enable combinations of tense and aspect. The following table gives a grammatical term and shows an example for each combination. The table assumes a domain vocabulary has a verb concept "company$_1$ *trades with* company$_2$". The table shows semantic concepts of tense and aspect using English syntax for illustration purposes only. Different natural languages use different syntaxes to express these semantics. Some natural languages do not distinguish each combination shown in the table. Annex E contains an informative formal analysis of English language syntax for tense and aspect.

Table 7.1 - Language Tense and Aspect

		Aspect			
		Simple	Progressive	Perfect	Progressive & Perfect
Tense	Past	past simple company x *traded with* company y	past progressive company x *was trading with* company y	past perfect, pluperfect company x *had traded with* company y	pluperfect progressive company x *had been trading with* company y
	Present	present simple company x *trades with* company y	present progressive company x *is trading with* company y	present perfect company x *has traded with* company y	present perfect progressive company x *has been trading with* company y
	Future	future simple company x *will trade with* company y	future progressive company x *will be trading with* company y	future perfect company x *will have traded with* company y	future perfect progressive company x *will have been trading with* company y

These combinations can be employed in business rules, as shown in these examples. They presume a domain vocabulary verb concept "company$_1$ *merges with* company$_2$".

1. "If some company$_1$ *merged with* the company x …" – asking whether a merger happened in the past, independent of whether the trading is ongoing, completed, or both.
2. "If some company$_1$ *was merging with* the company x …" – asking whether a merger was continuing over some time in the past."
3. "If some company$_1$ *will have merged with* the company x …" – asking whether a merger will be accomplished in the future.
4. "If some company$_1$ *will have been merging with* the company x …" – asking whether a completed merger will be ongoing in the future.

One intended use case for these many combinations is annotation of existing text, as in [TimeML].

Clause 10.3, "Uses of Time" provides vocabulary for formulating tenses and aspects, and describes how these may be combined in rules.

7.12 Domain Vocabularies and Time

This specification provides foundational date and time concepts that are intended for use in domain-specific business vocabularies and rules. Annex C gives a complete example. This sub clause shows an abbreviated example in order to introduce how a domain vocabulary can build on this Date-Time Vocabulary.

Consider the example of a contract that has a "start date", a "contract length", a "contract term", and a "payment schedule". A business vocabulary might specify these as follows:

Example Vocabulary

General Concept: terminological dictionary

Language: English

contract

Definition: Agreement between two companies for one to provide goods or services, and for the other to pay for those goods or services

start date

General Concept: calendar day

Note: The granularity of a domain vocabulary time concept is defined via the time point kind. Defining 'start date' as a calendar day means that the granularity of 'start date' is 'day' rather than 'week' or 'month', etc.

Note: Domain vocabulary time concepts should be defined as kinds of 'time point' or 'duration', rather than 'time coordinate' or 'duration value'. Actual 'time points' and 'durations' can be specified as definite descriptions as well as 'time coordinates' and 'duration values'.

contract *has* start date

contract length

General Concept: duration

Necessity: The granularity *of* 'contract length' *is* 'day'.

contract *has* contract length

contract term

Definition: Time interval during which the goods should be delivered or the services provided.

Necessity: The time interval *of* a contract *is from* the start date *of* the contract *for* the contract length.

contract *has* contract term

payment schedule

Definition: schedule *for* contract payments in which the time span *is* the contract term, and the repeat interval *is* 1 month

contract *has* payment schedule

contract payment

Definition: amount to be paid according to the payment schedule

contract *has* contract payment

A business rule example might be:

It is obligatory that a contract payment *be paid on* each time table entry *of* the payment schedule.

The example is simplified since it does not specify all the details that would exist in a real contract. For example, it does not indicate who makes the payment or who receives the payment, nor does it allow for payments other than monthly. But it does illustrate some basic ideas:

1. Defining domain vocabulary concepts that make use of time points (start date), durations (contract length), time intervals (contract term), and schedules (payment schedule).
2. Using Definitions (start date, contract term, payment schedule) and Necessities (contract term) to precisely capture the semantic meaning of domain concepts.
3. Specifying business rules that build upon this Date-Time Vocabulary and domain vocabularies to model business requirements.

Consider a business rule such as "It is obligatory that the contract length *of* each contract *is less than* 1 year." Notice that it compares 'contract length' to '1 year'. It does not quantify over 'year' because time is a mass noun concept. In contrast, a rule such as "It is obligatory that each rental *has* at most 3 additional drivers" uses quantification because 'additional driver' is a countable noun concept. Mass noun concepts are measured (possibly in fractional units of measure) while countable noun concepts are counted in whole units.

7.13 Enabling Other Calendars

The world has many different time-keeping and calendar systems. Specialized business calendars include fiscal calendars, tax calendars, and manufacturing calendars. Examples of historical, religious, and cultural calendars include the Julian calendar, various lunar calendars, and the 14-year calendar cycle of some Asian nations. Examples of time-keeping systems are those based on mariners' "bells", and religious "vespers".

This specification defines vocabularies for the standard, globally recognized "Universal Date Coordinated" (UTC) time system, and the Gregorian Calendar. In addition, this specification provides a Time Infrastructure Vocabulary that enables others to define business domain-specific, cultural, religious, or historical calendars and time schemas. The Time of Day Vocabulary and Gregorian Calendar Vocabulary show how time and calendar systems can be defined using the foundational concepts of the Time Infrastructure Vocabulary. Specifying time systems and calendars in terms of the foundational concepts of the Time Infrastructure Vocabulary enables conversions between different calendars and different time keeping schemas.

8 Time Infrastructure (normative)

Many time schemes and calendars are in use to support a variety of business needs, and due to historical, cultural, and religious traditions. The “Time Infrastructure” vocabulary provides a foundation for defining any time keeping or calendar system. Relating different time and calendar schemes to each other is made possible by using the foundational concepts provided in this clause.

Time Infrastructure Vocabulary

General Concept: terminological dictionary

Language: English

Various vocabularies, standards, and other publications that are referenced in the SBVR aspects of this specification are formally named as SBVR "individual constants" here.

IEC 60050-111

Definition: The standard of the International Electrotechnical Committee, International Electrotechnical Vocabulary, number-60050 Chapter 111, named: *Physics and Chemistry*, Edition 2.0, 1996-07

ISO 8601

Definition: The standard of the International Standards Organization (ISO), number 8601, named: *Data elements and interchange formats – Information interchange – Representation of Dates and Times*, Third edition, December 1, 2004

NODE

Definition: The publication named: *New Oxford Dictionary of English*

8.1 The Time Axis and Time Intervals

The principal concept in this subclause is time interval. This concept is used to define many of the business terms that are specified in other clauses of this specification. Formally, time interval is a primitive concept – an intuitive notion that does not have a mathematical definition. Its properties are defined by a set of axioms that are presented here as SBVR definitions and Necessities with matching CLIF and OCL statements. Much of this clause is the presentation of those axioms.

Time Axis

Dictionary Basis: IEC 60050-111 (‘time axis’)

Dictionary Basis: IEC 8601 (2.1.1, ‘time axis’)

General Concept: mathematical representation of the succession in time of instantaneous events along a unique axis

Source: NODE (‘time’)

Definition: the indefinite continued progress of existence and events in the past, present, and future, regarded as a continuum

Necessity: There exists exactly one Time Axis.

Note: The above necessity is questionable in light of the theory of relativity, but relativistic effects are not considered in this model. Some applications need to take these effects into account, e.g., GPS, in which the clocks in satellites are adjusted on the ground to compensate for relativistic shifts in their rates in orbit, due to the lower gravitational field in orbit (+) and orbital motion (-).

Note: Time Axis is the conceptual time dimension.

Note: "Time" could be a synonym of Time Axis, but "time" is often confused with other concepts, such as duration and time of day.

time interval

Definition: segment of the time axis, a location in time

Example: The lifetime of Henry V.

Example: The day whose Gregorian calendar date is September 11, 2001.

8.1.1 The Whole-Part Relationship Among Time Intervals

The mereological principles described in Annex D.4 apply to time intervals.

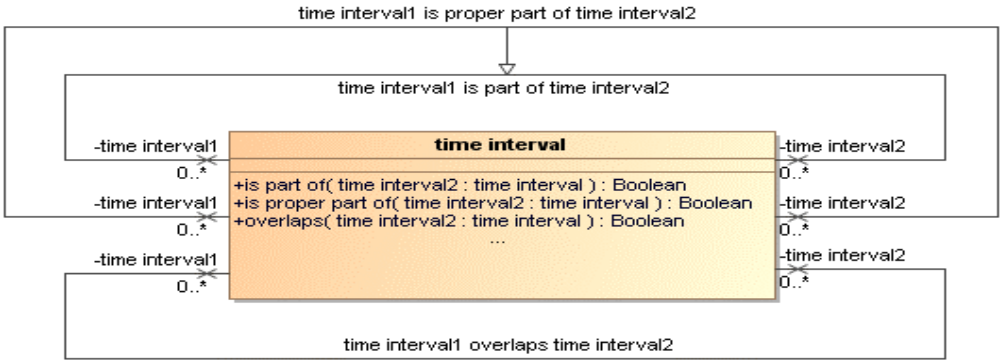

Figure 8.1 - Mereology as Applied to Time Intervals

time interval$_1$ *is part of* time interval$_2$

Synonymous Form: time interval$_2$ *includes* time interval$_1$

Definition: Time interval$_2$ is a component of time interval$_1$. Every instant in time interval$_1$ is also in time interval$_2$. Everything that happens in time interval$_1$ happens in time interval$_2$

Note: Like the concept time interval itself, this relationship is also primitive – intuitive. It is a mathematical ordering of time intervals by containment.

Note: This relationship is based on the mereological verb concept 'part *is part of* whole' (Annex D.4). All the axioms cited there for 'part *is part of* whole' apply to 'time interval$_1$ *is part of* time interval$_2$'.

Note: The axioms of reflexivity, antisymmetry, and transitivity (Annex D.4) make 'time interval$_1$ *is part of* time interval$_2$' a partial ordering relationship on time intervals. The relationship is *partial* because two arbitrary time intervals might be disjoint or might overlap, so that there is no part-whole relationship between them.

time interval$_1$ *overlaps* time interval$_2$

Synonymous Form: time interval$_2$ *overlaps* time interval$_1$

Note: This relationship is based on the mereological verb concept 'part$_1$ *overlaps* part$_2$' (Annex D.4). See the definition of that concept for details.

time interval$_1$ *is a proper part of* time interval$_2$

Note: This relationship is based on the mereological verb concept 'part *is a proper part of* whole' (Annex D.4). See the definition of that concept for details. See also the axiom of supplementation given in that Annex.

Note: A proper part is a part that is not the whole.

8.1.2 The Temporal Ordering Relationship

A fundamental property of time intervals is the totally ordered *'is before'* relationship, which defines temporal ordering.

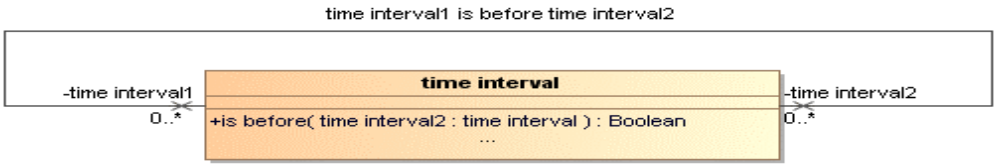

Figure 8.2 - Temporal Ordering

time interval$_1$ *is before* time interval$_2$

Synonymous Form: time interval$_2$ *is after* time interval$_1$

Synonymous Form: time interval$_1$ **≤** time interval$_2$

Synonymous Form: time interval$_2$ **≤** time interval$_1$

Definition: time interval$_1$ ends before/when time interval$_2$ starts

Example: The time interval identified by 2010 is before the time interval identified by 2011.

Note: This relationship is also primitive – intuitive. It is a mathematical ordering of time intervals by position on the Time Axis. *Is before* captures the intuition of the direction of time, of past and future: if x is before y, then y is in the future relative to x and x is in the past relative to y.

Note: The actual determination of the ordering of time intervals may be based on direct observation, on calendar knowledge, on historical knowledge, or on practical knowledge. One can see the order in which two vehicles enter an intersection and infer the corresponding facts about the time intervals involved (observation). One can know from calendar rules that November 11, 1918 was before September 1, 1939. One can know from the reports of others (historical knowledge) that railroads were in use for many years before automobiles first appeared. Knowing that every airplane takes off before it lands (practical knowledge), and that a particular airplane has taken off and landed, one can infer that the time interval of the takeoff was before the time interval of the landing. And, of course, these knowledge elements can be mixed in determining time interval ordering. When such knowledge elements are formalized as facts and rules in an ontology, the inferences about the ordering of time intervals can be automated.

Note: The following axioms define the properties of this primitive concept.

Axiom: time interval$_1$ *is before* time interval$_2$ can only be true of time intervals that do not overlap.

Necessity:	If a time interval$_1$ *overlaps* a time interval$_2$, then the time interval$_1$ *is* not *before* the time interval$_2$.
CLIF Axiom:	(forall ((t1 "time interval") (t2 "time interval")) (if ("time interval1 overlaps time interval2" t1 t2) (not ("time interval1 is before time interval2" t1 t2))))
OCL Constraint:	context "time interval" inv: "time interval".allInstances-->forAll(t2 | self.overlaps(t2) implies not self.before(t2))

Corollary:

Necessity:	If a time interval$_1$ *overlaps* a time interval$_2$, then the time interval$_1$ *is* not *after* the time interval$_2$.
Note:	This follows from the fact that 'time interval$_1$ *overlaps* time interval$_2$' is a synonymous form of 'time interval$_2$ *overlaps* time interval$_1$'.

Corollary:

Necessity:	If a time interval$_1$ does not *overlap* a time interval$_2$, then the time interval$_1$ *is before* the time interval$_2$ or the time interval$_2$ *is before* the time interval$_1$.
CLIF Axiom:	(forall ((t1 "time interval") (t2 "time interval")) (if (not ("time interval1 overlaps time interval2" t1 t2)) (or ("time interval1 is before time interval2" t1 t2) ("time interval1 is before time interval2" t2 t1))))
OCL Constraint:	context "time interval" inv: "time interval".allInstances-->forAll(t2 | not self.overlaps(t2) implies (self.before(t2) or t2.before(self)))

Corollary (*irreflexivity*):

Necessity:	A given time interval *is* not *before* the time interval.
CLIF Axiom:	(forall ((t1 "time interval")) (not ("time interval1 is before time interval2" t1 t1)))
OCL Constraint:	context "time interval" inv: not self.before(self)

Axiom of *asymmetry*: No time interval is both *before* and *after* the same time interval.

Necessity:	If a time interval$_1$ *is before* a time interval$_2$, then the time interval$_2$ *is* not *before* the time interval$_1$.

CLIF Axiom:

```
(forall ((t1 "time interval") (t2 "time interval"))
  (if ("time interval1 is before time interval2" t1 t2)
     (not ("time interval1 is before time interval2"
          t2 t1))))
```

OCL Constraint:

```
context "time interval"
inv: "time interval".allInstances-->forAll(t2 |
    self.before(t2)
      implies not t2.before(self))
```

Corollary (totality): For any two time intervals *t1* and *t2*, exactly one of the following is true:

- *t1 overlaps t2*
- *t1 is before t2*
- *t2 is before t1*

Necessity: Each time interval$_1$ *overlaps* each time interval$_2$ and time interval$_1$ *is* not *before* time interval$_2$ and time interval$_2$ *is* not *before* time interval$_1$, or time interval$_1$ *is before* time interval$_2$ and time interval$_1$ *does* not *overlap* time interval$_2$ and time interval$_2$ *is* not *before* time interval$_1$, or time interval$_2$ *is before* time interval$_1$ and time interval$_1$ *does* not *overlap* time interval$_2$ and time interval$_1$ *is* not *before* time interval$_2$.

CLIF Axiom:

```
(forall ((t1 "time interval") (t2 "time interval))
  (or
    (and
     ("time interval1 overlaps time interval2" t1 t2)
     (not "time interval1 is before time interval2" t1 t2))
     (not "time interval1 is before time interval2"
          t2 t1))
    (and
     ("time interval1 is before time interval2" t1 t2)
     (not "time interval1 overlaps time interval2" t1 t2))
     (not "time interval1 is before time interval2"
          t2 t1))
    (and
     ("time interval1 is before time interval2" t2 t1)
     (not "time interval1 overlaps time interval2" t1 t2))
     (not "time interval1 is before time interval2"
          t1 t2))))
```

OCL Constraint:

```
context "time interval"
inv: "time interval".allInstances-->forAll(t2 |
    (self.overlaps(t2) and not self.before(t2) and
      not t2.before(self))
   or (self.before(t2) and not self.overlaps(t2) and
      not t2.before(self))
   or (t2.before(self) and not self.overlaps(t2) and
      not self.before(t2)))
```

Axiom of *transitivity*: Every time interval that *is before* a given time interval is also *before* every time interval that is after the given time interval.

Necessity:	If a time interval$_1$ *is before* a time interval$_2$ and the time interval$_2$ is before a time interval$_3$ then the time interval$_1$ *is before* the time interval$_3$.
CLIF Axiom:	(forall ((t1 "time interval") (t2 "time interval") (t3 "time interval")) (if (and ("time interval1 is_before time interval2" t1 t2) ("time interval1 is_before time interval2" t2 t3)) ("time interval1 is_before time interval2" t1 t3))))
OCL Constraint:	context "time interval" inv: "time interval".allInstances-->forAll(t2 | self.before(t2) and "time interval".allInstances-->forAll(t3 | t2.before(t3) implies self.before(t3)))

The preceding 3 axioms specify that 'time interval$_1$ *is before* time interval $_2$' is anti-reflexive, weakly antisymmetric, and transitive. The relationship does *not* apply to all pairs of time intervals. This characterizes a kind of partial ordering on time intervals.

8.1.3 The Allen Relations

In a 1983 paper [Allen], James F. Allen asserted that there are exactly thirteen ways in which an ordered pair of time intervals can be related. His Figure 2, showing these relationships, is reproduced below.

Relation	Symbol	Symbol for Inverse	Pictoral Example
X *before* Y	<	>	XXX YYY
X *equal* Y	=	=	XXX YYY
X *meets* Y	m	mi	XXXYYY
X *overlaps* Y	o	oi	XXX YYY
X *during* Y	d	di	XXX YYYYYY
X starts Y	s	si	XXX YYYYY
X finishes Y	f	fi	XXX YYYYY

FIGURE 2. The Thirteen Possible Relationships.

Figure 8.3 - Allen's Original Diagram of the 13 Time Relationships

According to Thomas Alspaugh [Alspaugh], these relations are *distinct* ("because no pair of definite intervals can be related by more than one of these relationships"), *exhaustive* ("because any pair of definite intervals are described by one of the relations"), and *qualitative*, rather than *quantitative*, ("because no numeric time spans are considered").

The word '*properly*' is used in the terms for some of the Allen relations below, in order to distinguish those relations from the more general relations defined in 8.1.1 and 8.1.2. In each case of terminology clash, the Allen's term is narrower. The business use of the general term – before, after, part of, includes, during, overlaps – almost always means the more general relationship.

The Allen relations are *independent*: none is entailed by another and none is defined in terms of the others. They are, however, all defined here in terms of the two fundamental relationships: '*part of*' and '*before*'.

The '*properly before*' and '*meets*' relations are mutually exclusive. The primitive relationship '*before*' subsumes both. Allen's '*before*' concept is designated here as '*properly before*' to indicate there is necessarily an intervening time interval.

The '*properly overlaps*' relation distinguishes the case in which there is a part of each time interval that *is not* a part the other from all the cases in which one time interval is entirely a part of the other. The general '*overlaps*' relation subsumes all of them. '*Properly overlaps*' describes the first time interval as starting and ending earlier than the second, whereas '*is properly overlapped by*' describes the first time interval as starting and ending later.

The '*properly during*,' '*starts*', and '*finishes*' relationships are mutually exclusive. The general '*part of*' relationship subsumes all of them. They are distinguished by the temporal relationship of the included time interval to the supplementary parts of the whole.

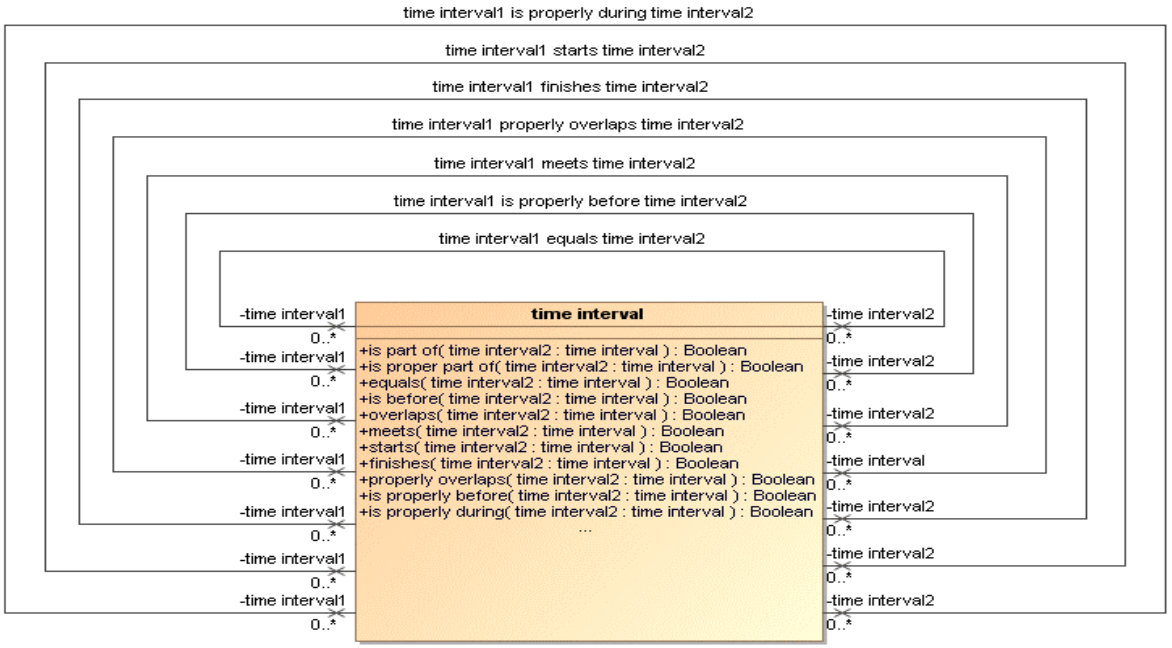

Figure 8.4 - UML Diagram of Allen Relations

time interval$_1$ *is properly before* time interval$_2$

Synonymous Form: time interval$_2$ *is properly after* time interval$_1$

Definition: the time interval$_1$ *is before* the time interval$_2$ and the time interval$_1$ *is before* a time interval$_3$ and the time interval$_3$ *is before* the time interval$_2$

CLIF Definition:

```
(forall ((t1 "time interval") (t2 "time interval"))
 (iff ("time interval1 is properly before time interval2" t1 t2)
    (and ("time interval1 is before time interval2" t1 t2)
```

(exists ((t3 "time interval"))
(and ("time interval1 is before time interval2" t1 t3)
("time interval1 is before time interval2" t3 t2)
)))))

OCL Definition:
context "time interval"
def: "time interval1 is properly before time interval2"
(t1: "time interval", t2: "time interval")
: Boolean =
t1.before(t2) and
"time interval".allInstances-->exists(t3 |
t1.before(t3) and t3.before(t2))

Example: 2009 *is properly before* 2011

time interval$_1$ *equals* time interval$_2$

Synonymous Form: time interval$_1$ *is the same as* time interval$_2$

Synonymous Form: time interval$_1$ = time interval$_2$

Definition: the time interval$_1$ *is part of* the time interval$_2$ and the time interval$_2$ *is part of* the time interval$_1$

CLIF Definition:
(forall ((t1 "time interval") (t2 "time interval"))
(iff ("time interval1 equals time interval2" t1 t2)
(and ("time interval1 is part of time interval2" t1 t2)
("time interval1 is part of time interval2" t2 t1))

OCL Definition:
context "time interval"
def: "time interval1 equals time interval2"
(t1: "time interval, t2: "time interval)
: Boolean =
t1."part of" (t2) and t2."part of"(t1)

Note: That is, the mereology axiom of antisymmetry in Annex D.4 is really the formal definition of '*equals*.' Two time intervals are equal if and only if each is part of the other.

Note: SBVR uses the verb *is* for this relationship, but the equals relationship here is a specialization of 'thing *is* thing' for time intervals. Two time intervals are *equal* if they share particular properties of time interval, and the definition of *equal* does not involve properties that are suitable for a reference scheme.

Example: January 2011 *through* December 2011 *equals* 2011

time interval$_1$ *meets* time interval$_2$

Synonymous Form: time interval$_2$ *is met by* time interval$_1$

Synonymous Form: time interval$_1$ *immediately precedes* time interval$_2$

Synonymous Form: time interval$_2$ *immediately follows* time interval$_1$

Definition: the time interval$_1$ *is before* the time interval$_2$ and the time interval$_1$ *is* not *before* a time interval$_3$ that *is before* the time interval$_2$

CLIF Definition:
(forall ((t1 "time interval") (t2 "time interval"))
(iff ("time interval1 meets time interval2" t1 t2)
(and ("time interval1 is before time interval2" t1 t2)
(not (exists (t3 "time interval"))
(and ("time interval1 is before time interval2"

t1 t3)
("time interval1 is before time interval2"
t3 t2))))))

OCL Definition:
```
context "time interval"
def: "time interval1 meets time interval2"
    (t1: "time interval", t2: "time interval")
    : Boolean =
    t1.before(t2) and
    not "time interval".allInstances-->exists(t3 |
        t1.before(t3) and t3.before(t2))
```

Example: 2009 *meets* 2010

time interval$_1$ *properly overlaps* time interval$_2$

Synonymous Form: time interval$_2$ *is properly overlapped by* time interval$_1$

Definition: the time interval$_1$ *overlaps* the time interval$_2$ and a time interval$_3$ *is a proper part of* the time interval$_1$ and the time interval$_3$ *is before* the time interval$_2$

CLIF Definition:
```
(forall ((t1 "time interval") (t2 "time interval"))
 (iff ("time interval1 properly_overlaps time interval2" t1 t2)
    (and ("time interval1 overlaps time interval2" t1 t2)
       (exists ((t3 "time interval"))
         (and ("time interval1 is proper part of time
              interval2" t3 t1)
            ("time interval1 is before time interval2"
             t3 t2))))))
```

OCL Definition:
```
context "time interval"
def: "time interval1 properly overlaps time interval2"
    (t1: "time interval", t2: "time interval")
    : Boolean =
    t1.overlaps(t2) and
    "time interval".allInstances-->exists(t3 |
      t3."is a proper part of"(t1) and t3.before(t2))
```

Example: July 2010 *through* February 2011 *properly overlaps* January 2011 *through* March 2011

time interval$_1$ *is properly during* time interval$_2$

Synonymous Form: time interval$_2$ *properly includes* time interval$_1$

Definition: the time interval$_1$ *is a proper part of* the time interval$_2$ and a time interval$_3$ *is a proper part of* the time interval$_2$ and a time interval$_4$ *is a proper part of* the time interval$_2$ and the time interval$_3$ *is before* the time interval$_1$ and the time interval$_1$ *is before* the time interval$_4$

CLIF Definition:
```
(forall ((t1 "time interval") (t2 "time interval"))
 (iff ("time interval1 is properly during time interval2" t1 t2)
    (and ("time interval1 is proper part of time interval2"
         t1 t2)
       (exists ((t3 "time interval") (t4 "time interval"))
         (and ("time interval1 is proper part of time
              interval2" t3 t2)
            ("time interval1 is proper part of time
```

```
                    interval2" t4 t2)
                  ("time interval1 is before time interval2"
                   t3 t1)
                  ("time interval1 is before time interval2"
                   t1 t4))))))
```

OCL Definition:

```
context "time interval"
def: "time interval1 is properly during time interval2"
    (t1: "time interval", t2: "time interval")
    : Boolean =
    t1."is a proper part of"(t2) and
    "time interval".allInstances-->exists(t3 |
      "time interval".allInstances-->exists(t4 |
        t3."is a proper part of"(t2) and
        t4."is a proper part of"(t2) and
        t3.before(t1) and
        t1.before(t4)))
```

Example: July 2010 *is properly during* 2010

time interval$_1$ *starts* time interval$_2$

Synonymous Form: time interval$_2$ *is started by* time interval$_1$

Definition: time interval$_1$ *is a proper part of* time interval$_2$ and there exists no time interval$_3$ that *is a proper part of* time interval$_2$ and that *is before* time interval$_1$

CLIF Definition:

```
(forall ((t1 "time interval") (t2 "time interval"))
 (iff ("time interval1 starts time interval2" t1 t2)
     (and ("time interval1 is proper part of time interval2"
           t1 t2)
          (not (exists ((t3 "time interval"))
                (and ("time interval1 is proper part of time
                       interval2" t3 t2)
                     ("time interval1 is before time interval2"
                      t3 t1))))))
```

OCL Definition:

```
context "time interval"
def: "time interval1 starts time interval2"
    (t1: "time interval", t2: "time interval")
    : Boolean =
    t1."is a proper part of"(t2) and
    not "time interval".allInstances-->exists(t3 |
        t3."is a proper part of"(t2) and
        t3.before(t1))
```

Example: January 2010 *starts* 2010

time interval$_1$ *finishes* time interval$_2$

Synonymous Form: time interval$_2$ *is finished by* time interval$_1$

Definition: time interval$_1$ *is a proper part of* time interval$_2$ and there exists no time interval$_3$ that *is a proper part of* time interval$_2$ and that *is after* time interval$_1$

CLIF Definition:

```
(forall ((t1 "time interval") (t2 "time interval"))
 (iff ("time interval1 finishes time interval2" t1 t2)
```

```
(and ("time interval1 is proper part of time interval2"
     t1 t2)
   (not (exists ((t3 "time interval"))
        (and ("time interval1 is proper part of time
              interval2" t3 t2)
             ("time interval1 is before time interval2"
              t1 t3))))))
```

OCL Definition:

```
context "time interval"
def: "time interval1 finishes time interval2"
   (t1: "time interval", t2: "time interval")
   : Boolean =
   t1."is a proper part of"(t2) and
   not "time interval".allInstances-->exists(t3 |
      t3."is a proper part of"(t2) and
      t1.before(t3))
```

Example: December 2010 *finishes* 2010

8.1.4 Additional Time Interval Relationships

As described in [Alspaugh], the basic Allen relationships can be combined in 2^{13} (8192) ways. This sub clause defines a few of these "combination" relationships that have particular value to everyday and business uses.

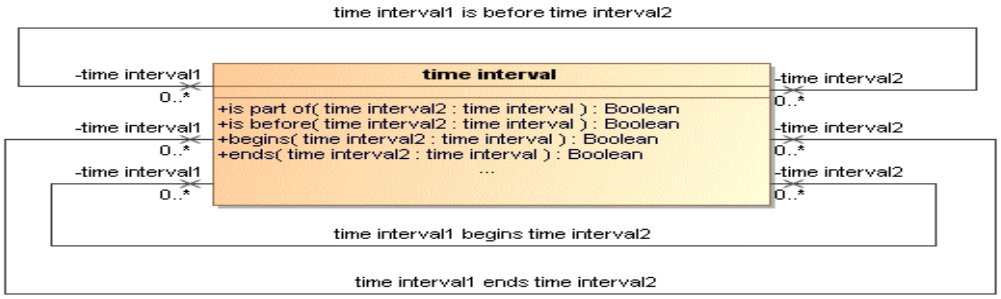

Figure 8.5 - Additional Time Interval Relationships

time interval$_1$ *precedes* time interval$_2$

Synonymous Form: time interval$_2$ *is preceded by* time interval$_1$

Synonymous Form: time interval$_2$ *follows* time interval$_1$

Synonymous Form: time interval$_1$ *is followed by* time interval$_2$

Definition: the time interval$_1$ *is properly before* the time interval$_2$ or the time interval$_1$ *meets* the time interval$_2$

CLIF Definition:

```
(forall ((t1 "time interval") (t2 "time interval"))
 (iff ("time interval1 precedes time interval2" t1 t2)
    (or ("time interval1 is properly before time interval2"
         t1 t2)
        ("time interval1 meets time interval2" t1 t2))))
```

OCL Definition:
```
context "time interval"
def: "time interval1 precedes time interval2"
    (t1: "time interval", t2: "time interval")
    : Boolean =
    t1."is properly before"(t2) or t1.meets(t2)
```

Example: 2009 *precedes* 2011

Example: 2009 *precedes* 2010

time interval$_1$ *begins* time interval$_2$

Synonymous Form: time interval$_2$ *is begun by* time interval$_1$

Definition: the time interval$_1$ *equals* the time interval$_2$ or the time interval$_1$ *meets* the time interval$_2$ or the time interval$_1$ *properly overlaps* the time interval$_2$ or the time interval$_1$ *starts* the time interval$_2$

CLIF Definition:
```
(forall ((t1 "time interval") (t2 "time interval"))
 (iff ("time interval1 begins time interval2" t1 t2)
     (or ("time interval1 equals time interval2" t1 t2)
        ("time interval1 meets time interval2" t1 t2)
        ("time interval1 properly overlaps time interval2"
         t1 t2)
        ("time interval1 starts time interval2" t1 t2))))
```

OCL Definition:
```
context "time interval"
def: "time interval1 begins time interval2"
    (t1: "time interval", t2: "time interval")
    : Boolean =
    t1.equals(t2) or
    t1.meets(t2) or
    t1."properly overlaps"(t2)
    t1.starts(t2)
```

Example: 2010 *begins* 2010

Example: 2009 *begins* 2010

Example: December 2009 *through* January 2010 *begins* 2010

Example: January 2010 *begins* 2010

time interval$_1$ *ends* time interval$_2$

Synonymous Form: time interval$_2$ *is ended by* time interval$_1$

Definition: the time interval$_1$ *equals* the time interval$_2$ or the time interval$_2$ *meets* the time interval$_1$ or the time interval$_2$ *properly overlaps* the time interval$_1$ or the time interval$_1$ *finishes* the time interval$_2$

CLIF Definition:
```
(forall ((t1 "time interval") (t2 "time interval"))
 (iff ("time interval1 ends time interval2" t1 t2)
     (or ("time interval1 equals time interval2" t1 t2)
        ("time interval1 meets time interval2" t2 t1)
        ("time interval1 properly overlaps time interval2"
         t1 t2)
        ("time interval1 finishes time interval2" t1 t2))))
```

OCL Definition:
```
context "time interval"
def: "time interval1 ends time interval2"
    (t1: "time interval", t2: "time interval")
    : Boolean =
    t1.equals(t2) or
    t2.meets(t1) or
    t2."properly overlaps"(t1)
    t1.finishes(t2)
```

Example: 2010 *ends* 2010

Example: 2010 *ends* 2009

Example: December 2009 *through* January 2010 *ends* 2009

Definition: December 2010 *ends* 2010

time interval$_1$ *starts before* time interval$_2$

Definition: time interval$_1$ *is before* time interval$_2$ *or* time interval$_1$ *meets* time interval$_2$ *or* time interval$_1$ *properly overlaps* time interval$_2$ or time interval$_2$ *is properly during* time interval$_1$

Example: 2009 *starts before* 2010

Example: 2010 *starts before* February 2010

time interval$_1$ *ends after* time interval$_2$

Definition: time interval$_2$ *is before* time interval$_1$ *or* time interval$_2$ *meets* time interval$_1$ *or* time interval$_2$ *properly overlaps* time interval$_1$ or time interval$_2$ *is properly during* time interval$_1$

Example: 2010 *ends after* February 2010

8.1.5 Time Interval Sum

The above sub clauses introduce the basic time interval comparison relationships as well as several derived relationships that are commonly used in business. This sub clause describes the “sum” or “convex hull” of two time intervals. This basic relationship cannot be proved from the foregoing.

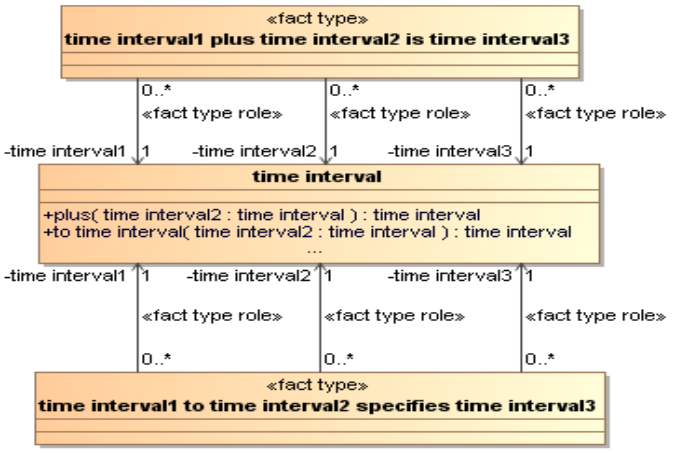

Figure 8.6 - Time Interval Sum

Axiom Sum: For any time intervals t1 and t2, there is a time interval t2+t2 with the following properties:

- if t1 *is before* t2 or t1 *properly overlaps* t2, t1+t2 *is started by* t1 and t1+t2 is finished by t2.
- if t1 *is after* t2 or t1 *is properly overlapped by* t2, t1+t2 *is started by* t2 and t1+t2 *is finished by* t1.
- if t1 *is part of* t2, t1+t2 = t2
- if t1 *includes* t2, t1+t2 = t1

Note that t1+t2 is well-defined. There are no other possible choices for the relationship between t1 and t2, and in the case t1 = t2 (which satisfies both of the last two conditions), t1 = t1+t2 = t2.

Corollary: For all time intervals t1 and t2, t1 *is part of* t1+t2 and t2 *is part of* t1+t2.

Corollary: For all time intervals t1, t2 and t3, such that t1 *is part of* t3 and t2 *is part of* t3, t1+t2 *is a part of* t3. That is, t1+t2 is the smallest time interval that includes both t and u.

Corollary: For any two time intervals t1 and t2, t1+t2 is unique.

This concept of 'sum' is generalized. It may be said to represent the 'convex hull' of the two intervals, and it may contain intervals that lie between them. It is particularly useful, however, when t1 meets t2 or t2 meets t1, i.e., in those cases where t1 and t2 are disjoint and there is no time interval between them.

time interval$_1$ *plus* time interval$_2$ *is* time interval$_3$

Synonymous Form: time interval$_1$ + time interval$_2$ = time interval$_3$

Synonymous Form: time interval$_3$ *is* time interval$_1$ *plus* time interval$_2$

Synonymous Form: time interval$_3$ = time interval$_1$ + time interval$_2$

Synonymous Form: time interval$_1$ *plus* time interval$_2$

Synonymous Form: time interval$_1$ + time interval$_2$

Synonymous Form: time interval$_1$ *through* time interval$_2$

Synonymous Form: *sum of* time interval$_1$ + time interval$_2$

Definition: if the time interval$_1$ *is before* the time interval$_2$ or the time interval$_1$ *properly overlaps* the time interval$_2$, then the time interval$_3$ *is started by* the time interval$_1$ and the time interval$_3$ *is finished by* the time interval$_2$

Definition: if the time interval$_1$ *is after* the time interval$_2$ or the time interval$_1$ *is properly overlapped by* the time interval$_2$, then the time interval$_3$ *is started by* the time interval$_2$ and the time interval$_3$ *is finished by* the time interval$_1$

Definition: if the time interval$_1$ *is part of* the time interval$_2$, then the time interval$_3$ *equals* the time interval$_2$

Definition: if the time interval$_1$ *includes* the time interval$_2$, then the time interval$_3$ *equals* the time interval$_1$

CLIF Definition:

```
(forall ((t1 "time interval") (t2 "time interval")
        (t3 "time interval"))
  (iff ("time interval1 plus time interval2 is
        time interval3" t1 t2 t3)
     (or
      (if (or
           ("time interval1 is before time interval2"
            t1 t2)
           ("time interval1 properly overlaps
             time interval2" t1 t2))
          (and
           ("time interval1 starts time interval2"
            t1 t3)
           ("time interval1 finishes time interval2"
            t2 t3)))
      (if (or
           ("time interval1 is before time interval2"
            t2 t1)
           ("time interval1 properly overlaps
             time interval2" t2 t1))
          (and
           ("time interval1 starts time interval2"
            t2 t3)
           ("time interval1 finishes time interval2"
            t1 t3)))
      (if ("time interval1 is part of
           time interval2" t1 t2)
          (= t2 t3))
      (if ("time interval1 is part of
           time interval2" t2 t1)
          (= t1 t3)))))
```

OCL Definition:

```
context "time interval"
def: "time interval1 plus
     time interval2 is time interval3"
     (t1: "time interval", t2: "time interval")
     : "time interval" =
   if t2.before(t1)
      or t1."properly overlaps"(t2)
```

```
then "time interval".allInstances-->exists(t3 |
      t3."started by"(t1)
      and t3."finished by"(t2)
      t3)
else if t1.before(t2)
  or t2."properly overlaps"(t1)
then "time interval".allInstances-->exists(t3 |
      t3."started by"(t2)
      and t3."finished by"(t1)
      t3)
else if t1."part of"(t2)
then t2
else t1
endif
```

Example: January 2010 *through* December 2010 *is* 2010

Corollary: For all time intervals t1 and t2, t1 *is part of* t1+t2 and t2 *is part of* t1+t2.

Necessity: A time interval$_1$ *plus* a time interval$_2$ *is* a time interval$_3$ and the time interval$_1$ *is part of* the time interval$_3$ and the time interval$_2$ *is part of* the time interval$_3$.

CLIF Axiom:

```
(forall ((t1 "time interval") (t2 "time interval"))
  (exists ((t3 "time interval"))
    (and
     ("time interval1 plus time interval2 is
       time interval3" t1 t2 t3)
     ("time interval1 is part of time interval2" t1 t3)
     ("time interval1 is part of time interval2" t2 t3))))
```

OCL Constraint:

```
context "time interval"
inv: "time interval".allInstances-->forAll(t2 |
       "time interval".allInstances-->exists(t3 |
         self."time interval1 plus time interval2 is time interval3"(t2, t3)))
```

Corollary: For all time intervals t1, t2 and t3, such that t1 *is part of* t3 and t2 *is part of* t3, t1+t2 is *part of* t3.

Necessity: If a time interval$_1$ *is part of* a time interval$_3$ and a time interval$_2$ *is part of* the time interval$_3$ then the time interval$_1$ *plus* the time interval$_2$ *is part of* the time interval$_3$.

CLIF Axiom:

```
(forall ((t1 "time interval") (t2 "time interval")
         (t3 "time interval"))
  (if (and
       ("time interval1 is part of time interval2" t1 t3)
       ("time interval1 is part of time interval2" t2 t3))
      ("time interval1 is part of time interval2"
       ("time interval1 plus time interval2 is
         time interval3" t1 t2)
       t3))))
```

OCL Constraint:

```
context "time interval"
inv: "time interval".allInstances-->forAll(t2 |
       "time interval".allInstances-->forAll(t3 |
       (self."time interval1 is part of time interval2"(t3)
```

```
and
t2. "time interval1 is part of time interval2"(t3))
implies self."time interval1 plus time interval2 is
        time interval3"(t2).
       "time interval1 is part of time
        interval2"(t3)))
```

Corollary: For any two time intervals t1 and t2, t1+t2 is unique.

Necessity: A time interval$_1$ *plus* a time interval$_2$ *is* exactly one time interval$_3$.

CLIF Axiom:

```
(forall ((t1 "time interval") (t2 "time interval"))
  (exists ((t3 "time interval"))
    (and
      ("time interval1 plus time interval2 is time interval3"
       t1 t2 t3)
      (forall ((t4 "time interval"))
        (if  ("time interval1 plus time interval2 is time
              interval3" t1 t2 t4)
             (= t4 t3))))))
```

OCL Constraint:

```
context "time interval"
inv: "time interval".allInstances-->forAll(t2 |
     "time interval".allInstances-->one(t4 | t4 =
      t1."time interval1 plus time interval2 is
         time interval3"(t2))
```

time interval$_1$ *to* time interval$_2$ *specifies* time interval$_3$

Synonymous Form: time interval$_1$ *to* time interval$_2$ *is* time interval$_3$

Synonymous Form: time interval$_1$ *to* time interval$_2$

Synonymous Form: time interval$_3$ *from* time interval1 *to* time interval$_2$

Definition: time interval$_1$ *is properly before* the time interval$_2$ and time interval$_3$ *is* time interval$_1$ *plus* a time interval$_4$ that *meets* time interval$_2$ and time interval$_1$ *meets* time interval$_4$

Definition: time interval$_1$ *meets* time interval$_2$ and time interval$_3$ *is* time interval$_1$

CLIF Definition:

```
(forall ((t1 "time interval") (t2 "time interval")
         (t3 "time interval"))
  (iff ("time interval1 to time interval2 specifies
         time interval3" t1 t2 t3)
       (or
        (and ("time interval1 is properly before
               time interval2" t1 t2)
             (exists ((t4 "time interval"))
               (and
               ("time interval1 meets time interval2"
                t4 t2)
               ("time interval1 meets time interval2"
                t1 t4)
               ("time interval1 plus time interval2 is
                 time interval3" t1 t4 t3)))
```

```
                (and ("time interval1 meets time interval2"
                        t1 t2)
                     (= t1 t3)))))
```

OCL Definition:

```
context "time interval"
def: "time interval1 to time interval2 specifies
     time interval3"
     (t1: "time interval", t2: "time interval")
     : "time interval" =
     ("time interval1 is properly before time interval2"
      (t1, t2) implies
       "time interval1 plus time interval2 is time
        interval3"("time interval".allInstances-->
                exists(t4 | t4.meets(t2) and
                        t1.meets(t4))-->at(1)))
     or
     (t1."time interval1 meets time interval2"(t2)
        implies t1)
```

Note: Contrast '*plus*' ('*through*') with '*to*.' '*Plus*' is inclusive of time interval$_2$, while '*to*' is exclusive of time interval$_2$.

Example: The time interval "year 2006" *to* the time interval "year 2007" *is* a time interval that has duration "1 year."

8.1.6 Time Interval Complement

The following start-complement and end-complement verb concepts construct the complementary time interval given a time interval that starts or ends a larger time interval. Note that a complementary time interval does not exist in the case where one time interval *is properly during* another time interval.

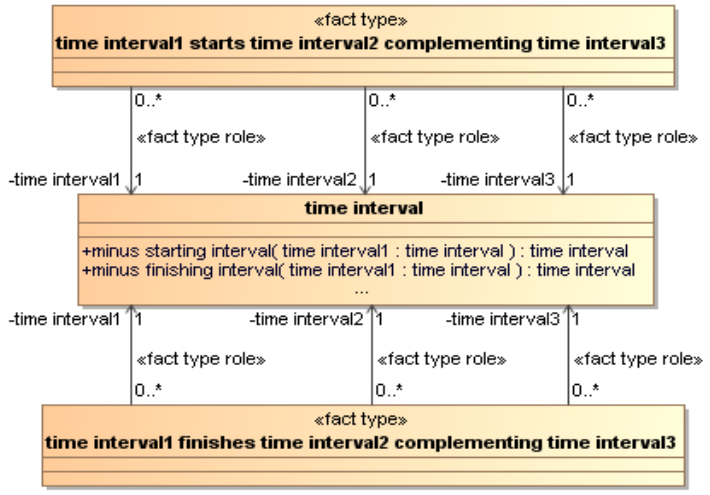

Figure 8.7 - Time Interval Complement

Axiom Start-complement: If t1 and t2 are time intervals and t1 *starts* t2, then there is a unique time interval t3 such that t3 *finishes* t2 and t1 *meets* t3.

time interval$_1$ *starts* time interval$_2$ *complementing* time interval$_3$

Definition: if the time interval$_1$ *starts* the time interval$_2$ then the time interval$_3$ *finishes* the time interval$_2$ and the time interval$_1$ *meets* the time interval$_3$

CLIF Definition:

```
(forall ((t1 "time interval") (t2 "time interval"))
 (exists (t3 "time interval"))
  (iff ("time interval1 starts time interval2 complementing
        time interval3" t1 t2 t3)
       (implies ("time interval1 starts time interval2"
                 t1 t2)
        (and
         ("time interval1 finishes time interval2" t3 t2)
         ("time interval1 meets time interval2" t1 t3))))))
```

OCL Definition:

```
context "time interval"
def: "time interval1 starts time interval2 complementing
     time interval3" (t2: "time interval",
                      t3: "time interval")
     : Boolean =
     self."time interval1 starts time interval2"(t2)
       implies t3.finishes(t2) and t1.meets(t3)
```

Example: January 2010 *starts* 2010 *complementing* February 2010 *through* December 2010

Corollary: For all time intervals t1, t2 and t3, such that t1 *starts* t2 *complementing* t3, and for all time intervals t4, such that t4 *is part of* t2 and t4 does not *overlap* t1, t4 *is part of* t3. That is, t3 is the largest time interval that *is part of* t2 but does not *overlap* t1.

Necessity: If a time interval$_1$ *starts* a time interval$_2$ *complementing* a time interval$_3$, then each time interval$_4$ that *is part of* the time interval$_2$ and that *does* not *overlap* the time interval$_1$ *is part of* the time interval$_3$.

CLIF Axiom:

```
(forall ((t1 "time interval") (t2 "time interval")
         (t3 "time interval"))
  (if ("time interval1 starts time interval2 complementing
        time interval3" t1 t2 t3)
      (forall ((t4 "time interval"))
        (if (and
             ("time interval1 is part of time interval2"
              t4 t2)
             (not ("time interval1 overlaps time
                    interval2" t4 t1)))
            ("time interval1 is part of time interval2"
             t4 t3)))))
```

OCL Constraint:

```
context "time interval"
inv: "time interval".allInstances-->forAll(t2 |
      "time interval".allInstances-->forAll(t3 |
     self."time interval1 starts time interval2
          complementing time interval3" t1 t2 t3)
```

```
implies "time interval".allInstances-->exists(t4 |
  (t4."time interval1 is part of time interval2"(t2)
   and not t4."time interval1 overlaps time
            interval2"(t1))
   implies t4."time interval is part of time
            interval2"(t3))))
```

Corollary: For any two time intervals t1 and t2 such that t1 *starts* t2 *complementing* some time interval t3, t3 is unique.

Necessity: If a time interval$_1$ *starts* a time interval$_2$ then the time interval$_1$ *starts* the time interval$_2$ *complementing* exactly one time interval$_3$.

CLIF Axiom:

```
(forall ((t1 "time interval") (t2 "time interval") (t3 "time interval"))
  (if ("time interval1 starts time interval2" t1 t2)
     (exists ((i integer))
       (and
         (count
           ("time interval1 starts time interval2 complementing time
             interval3" t1 t2) i)
         (= i 1)))))
```

OCL Constraint:

```
context "time interval"
inv: "time interval"-->forAll(t2 |
      self.starts(t2) implies
      self."time interval1 starts time interval2
           complementing time interval3"(t1)-->sum() = 1
```

Axiom End-complement: If t1 and t2 are time intervals and t1 *finishes* t2, then there is a unique time interval t3 such that t3 *starts* t2 and t1 *is met by* t3.

time interval$_1$ *finishes* time interval$_2$ *complementing* time interval$_3$

Definition: if the time interval$_1$ *finishes* the time interval$_2$ then the time interval$_3$ *starts* the time interval$_2$ and the time interval$_1$ *is met by* the time interval$_3$

CLIF Definition:

```
(forall ((t1 "time interval") (t2 "time interval"))
 (exists (t3 "time interval"))
  (iff ("time interval1 finishes time interval2
        complementing time interval3" t1 t2 t3)
       (if ("time interval1 finishes time interval2" t1 t2)
         (and
          ("time interval1 starts time interval2" t3 t2)
          ("time interval1 meets time interval2" t3 t1))))))
```

OCL Definition:

```
context "time interval"
def: "time interval1 finishes time interval2 complementing
     time interval3" (t2: "time interval",
                      t3: "time interval")
     : Boolean
     self."time interval1 finishes time interval2"(t2)
       implies t3.starts(t2) and t3.meets(t1)
```

Example: December 2010 *finishes* 2010 *complementing* January 2010 *through* February 2010

Corollary: For all time intervals t1, t2 and t3, such that t1 *finishes* t2 *complementing* t3, and for all time intervals t4, such that t4 *is part of* t2 and t4 does not *overlap* t1, t4 *is part of* t3. That is, t3 is the largest time interval that *is part of* t2 but does not *overlap* t1.

Necessity: If a time interval$_1$ *finishes* a time interval$_2$ *complementing* a time interval$_3$, then each time interval$_4$ that *is part of* the time interval$_2$ and that *does* not *overlap* the time interval$_1$ *is part of* the time interval$_3$.

CLIF Axiom:

```
(forall ((t1 "time interval") (t2 "time interval")
        (t3 "time interval"))
  (if ("time interval1 finishes time interval2 complementing
       time interval3" t1 t2 t3)
     (exists ((t4 "time interval"))
       (if (and
            ("time interval1 is part of time interval2"
              t4 t2)
            (not ("time interval1 overlaps time
                   interval2" t4 t1)))
          ("time interval1 is part of time interval2"
           t4 t3)))))
```

OCL Constraint:

```
context "time interval"
inv: "time interval".allInstances-->forAll(t2 |
      "time interval".allInstances-->forAll(t3 |
    self."time interval1 finishes time interval2
         complementing time interval3" t1 t2 t3)
    implies "time interval".allInstances-->exists(t4 |
     (t4."time interval1 is part of time interval2"(t2)
      and not t4."time interval1 overlaps time
              interval2"(t1))
       implies t4."time interval is part of time
                interval2"(t3))))
```

Corollary: For any two time intervals t1 and t2 such that t1 *finishes* t2 *complementing* some time interval t3, t3 is unique.

Necessity: If a time interval$_1$ *finishes* a time interval$_2$ then the time interval$_1$ *finishes* the time interval$_2$ *complementing* exactly one time interval$_3$.

CLIF Axiom:

```
(forall ((t1 "time interval") (t2 "time interval") (t3 "time interval"))
  (if ("time interval1 finishes time interval2" t1 t2 t3)
     (= 1
        (count
          ("time interval1 finishes time interval2 complementing"
            t1 t2)))))
```

OCL Constraint:

```
context "time interval"
inv: "time interval"-->forAll(t2 |
     self.finishes(t2) implies
     self."time interval1 starts time interval2
          complementing time interval3"(t1)-->sum() = 1
```

8.1.7 Time Interval Intersection

This verb concept generates the intersection of two overlapping **time intervals**.

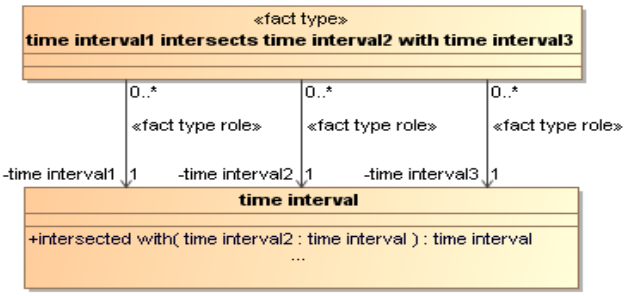

Figure 8.8 - Time Interval Intersection

Axiom Intersection: For any time intervals t1 and t2 such that t1 *overlaps* t2, there is a time interval t1*t2 that *finishes* t1 and that *starts* t2. We say that t1*t2 *intersects* t1 and t2.

time interval$_1$ *intersects* time interval$_2$ *with* time interval$_3$

Definition:	if the time interval$_1$ *overlaps* the time interval$_2$, then the time interval$_3$ *finishes* the time interval$_1$ and *starts* the time interval$_2$
CLIF Definition:	(forall ((t1 "time interval") (t2 "time interval") (t3 "time interval")) (iff ("time interval1 intersects time interval2 with time interval3" t1 t2 t3) (if ("time interval1 overlaps time interval2" t1 t2) (and ("time interval1 finishes time interval2" t3 t1) ("time interval1 starts time interval2" t3 t2)))))
OCL Definition:	context "time interval" def: "time interval1 intersects time interval2 with time interval3" (t2: "time interval", t3: "time interval") : Boolean = "time interval".allInstances-->forAll(t2 | self.overlaps(t2) implies "time interval".allInstances-->exists(t3 | t3.finishes(t1) and t3.starts(t2)))
Example:	January 2010 *through* June 2010 *intersects* March 2010 *through* August 2010 *with* March 2010 *through* June 2010

Corollary: For all time intervals t1, t2, and t4, such that t1 *overlaps* t2 and t4 *is part of* t1 and t4 *is part of* t2, t4 *is a part of* t1*t2. That is, t1*t2 is the largest time interval that *is part of* t1 and *part of* t2.

Necessity: If a time interval$_1$ *intersects* a time interval$_2$ *with* a time interval$_3$ and a time interval$_4$ *is part of* the time interval$_1$ and the time interval$_4$ *is part of* the time interval$_2$, then the time interval$_4$ *is part of* the time interval$_3$.

CLIF Axiom:

```
(forall ((t1 "time interval") (t2 "time interval")
        (t3 "time interval")(t4 "time interval"))
   (if (and
        ("time interval1 intersects time interval2 with
          time interval3" t1 t2 t3)
        ("time interval1 is part of time interval2"
          t4 t1)
        ("time interval1 is part of time interval2"
          t4 t2))
       ("time interval1 is part of time interval2" t4 t3))))
```

OCL Constraint:

```
self "time interval"
inv: "time interval".allInstances-->forAll(t2 |
     "time interval".allInstances-->forAll(t3 |
      "time interval".allInstances-->forAll(t4 |
       (self."time interval1 intersects time interval2
            with time interval3"(t2, t3)
        and
        t4."time interval1 is part of time interval2"(t1)
        and
        t4. "time interval1 is part of time interval2"(t1))
       implies
        t4."time interval1 is part of time interval2"(t3)
        )))
```

Corollary: For any two time intervals t1 and t2 such that t1 *overlaps* t2, t*u is unique.

Necessity: If a time interval$_1$ *overlaps* a time inteval$_2$, then the time interval$_1$ *intersects* a time interval$_2$ *with* exactly one time interval$_3$.

CLIF Axiom:

```
(forall ((t1 "time interval") (t2 "time interval"))
  (if ("time interval1 overlaps time interval2" t1 t2)
     (= 1
      (count
       ("time interval1 intersects time interval2"
         t1 t2)))))
```

OCL Constraint:

```
context "time interval"
inv: "time interval".allInstances-->forAll(t2 |
     self.overlaps(t2) implies
     self."time interval1 intersects time interval2 with
          time interval3"(t2)-->size() = 1
```

Corollary (Intervening): For all time intervals t1 and t2 such that t1 *is properly before* t2, there is a unique time interval t3 such that t1 *meets* t3 and t3 *meets* t2. The intervening time interval t3 is the *intersection* of the start-complement (t5) of t1+t2 (t4), and the end-complement of t1+t2 (t4).

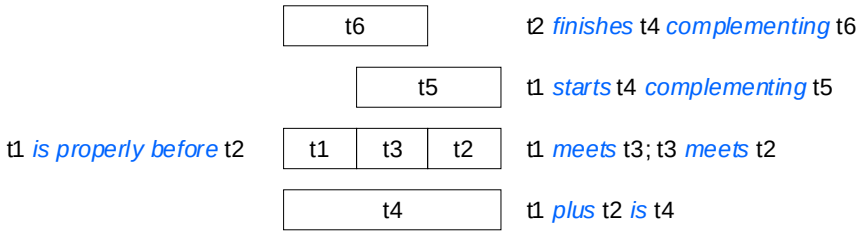

Figure 8.9 - Illustratiion of 'Intervening' Corollary

Necessity:

If a time interval$_1$ *is properly before* a time interval$_2$ then the time interval$_1$ *meets* a time interval$_3$ and the time interval$_3$ *meets* the time interval$_2$ and the time interval$_1$ *plus* the time interval$_2$ *is* a time interval$_4$ and the time interval$_1$ *starts* the time interval$_4$ *complementing* a time interval$_5$ and the time interval$_2$ *finishes* the time interval$_4$ *complementing* a time interval$_6$ and the time interval$_5$ *intersects* the time interval$_6$ *with* the time interval$_3$.

CLIF Axiom:

```
(forAll ((t1 "time interval") (t2 "time interval"))
  (if ("time interval1 is properly before
       time interval2" t1 t2")
     (exists ((t3 "time interval") (t4 "time interval")
           (t5 "time interval") (t6 "time interval"))
       (and
         ("time interval1 meets time interval2" t1 t3)
         ("time interval1 meets time interval2" t3 t2)
         ("time interval1 plus time interval2 is
           time interval3" t1 t2 t4)
         ("time interval1 starts time interval2
           complementing time interval3" t1 t4 t5)
         ("time interval1 finishes time interval2
           complementing time interval3" t2 t4 t6)
         ("time interval1 intersects time interval2
           with time interval3" t5 t6 t3)))))
```

OCL Constraint:

```
context "time interval"
inv: "time interval".allInstances-->forAll(t2 |
    self."time interval1 is properly before time interval2"
      (t2) implies
     "time interval".allInstances-->exists(t3 |
     "time interval".allInstances-->forAll(t4 |
     "time interval".allInstances-->forAll(t5 |
     "time interval".allInstances-->forAll(t6 |
     t1.meets(t3) and
     t3.meets(t2) and
     t1."time interval1 plus time interval2 is time
        interval3"(t2, t4) and
     t1."time interval1 starts time interval2
        complementing time interval3"(t4, t5) and
     t2."time interval1 finishes time interval2
```

complementing time interval3"(t4, t6) and
t5."time interval1 intersects time interval2 with
time interval3"(t6, t3))))))

8.2 Durations

A second foundational temporal concept is ‘duration,’ the amount of time in a time interval. This clause presents various properties of ‘duration’ and of the relationship between ‘duration’ and ‘time interval’.

duration

Synonym:	time
Definition:	base quantity of the International System of Quantities, used for measuring time intervals
Note:	Duration is a quantity kind, whose instances are quantities of time. Each duration is an equivalence class of particular durations: a duration equals all the measurements for the same amount of time.
Reference Scheme:	a precise atomic duration value that *quantifies* the duration
Source:	[ISO/IEC 80000-3]

8.2.1 Duration Ordering

‘Duration’ has relationships, '=', '≤', and '<' with the following properties. These relationships neither follow from nor entail the duration properties defined in the next clause.

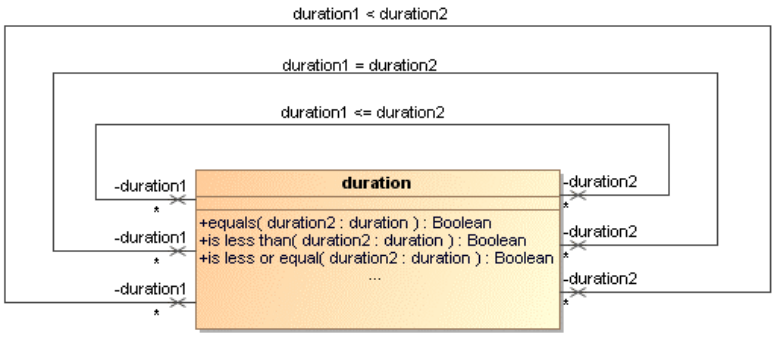

Figure 8.10 - Duration Ordering

Axiom O.1. Reflexive: If d1 is a duration, then d1 ≤ d1.

Axiom O.2. Total: If d1 and d2 are durations, then either d1 ≤ d2 or d2 ≤ d1.

Axiom O.3. Antisymmetric: If d1 and d2 are durations, and d1 ≤ d2 and d2 ≤ d1, then d1 = d2.

Axiom O.4. Transitive: If d1, d2, d3 are durations, and d1 ≤ d2 and d2 ≤ d3, then d1 ≤ d3.

The preceding four axioms, taken together, define a total ordering on ‘duration.’

Axiom O.5. Definition of <. If d1 and d3 are durations, d1 < d2 if d1 ≤ d2 and not d1 = d2

duration$_1$ ≤ duration$_2$

Synonymous Form:	duration$_2$ ≥ duration$_1$
Synonymous Form:	duration$_1$ *is less than or equal to* duration$_2$
Synonymous Form:	duration$_2$ *is greater than or equal to* duration$_1$
Definition:	A total ordering on quantities of time.
Note:	This is a primitive concept.
Example:	Two runners start a race at the same time. The duration of the run of one runner is less than or equal to the duration of the run of the other runner.

duration$_1$ = duration$_2$

Synonymous Form:	duration$_1$ *equals* duration$_2$
Definition:	duration$_1$ ≤ duration$_2$ and duration$_2$ ≤ duration$_1$
Example:	Two runners start and complete a race at the same time. The duration of the run of one runner is equal to the duration of the run of the other runner.

Axiom O.1: Reflexive: If d1 and d2 are durations, then d1 ≤ d2.

Necessity:	Each duration ≤ the duration.
CLIF Axiom:	(forall ((d duration)) (≤ d1 d1))
OCL Constraint:	context duration inv reflexive: self."duration1 ≤ duration2"-->contains(self)

Axiom O.2. Total: If d1 and d2 are durations, then either d1 ≤ d2 or d2 ≤ d1.

Necessity:	Each duration$_1$ ≤ each duration$_2$ or duration$_2$ ≤ duration$_1$.
CLIF Axiom:	(forall ((d1 duration) (d2 duration)) (or (≤ d1 d2) (≤ d2 d1)))

Issue: Not sure how to write this in OCL.

Axiom O.3. Antisymmetric: If d1 and d2 are durations, and d1 ≤ d2 and d2 ≤ d1, then d1 = d2.

Necessity:	If some duration$_1$ ≤ some duration$_2$ and the duration$_2$ ≤ the duration$_1$, then the duration$_1$ *equals* the duration$_2$.
CLIF Axiom:	(if (and (≤ d1 d2) (≤ d2 d1)) (= d1 d2)))
OCL Constraint:	context duration inv antisymmetric: self."duration1 ≤ duration2"-->forall(d2 | d2."duration1 ≤ duration2"-->contains(self) implies self = d2

Axiom O.4. Transitive: If d1, d2, d3 are durations, and d1 ≤ d2 and d2 ≤ d3, then d1 ≤ d3.

Necessity:	If some $duration_1$ ≤ some $duration_2$ and the $duration_2$ ≤ the $duration_3$ then the $duration_1$ ≤ the $duration_3$.
CLIF Axiom:	(forall ((d1 duration) (d2 duration) (d3 duration)) (if (and (≤ d1 d2) (≤ d2 d3)) (≤ d1 d3)))
CLIF Axiom:	context duration inv transitive: self."duration1 ≤ duration2"-->forAll(d2 | d2."duration1 ≤ duration2"-->forAll(d3 | self ≤ d3))

Axiom O.5. Definition of <: If d1 and d3 are durations, d1 < d2 iff d1 ≤ d2 and not d1 = d2

$duration_1$ < $duration_2$

Synonymous Form:	$duration_2$ > $duration_1$
Synonymous Form:	$duration_1$ *is less than* $duration_2$
Synonymous Form:	$duration_2$ *is greater than* $duration_1$
Definition:	$duration_1$ ≤ $duration_2$ and $duration_1$ *does* not *equal* $duration_2$
Example:	Two runners start a race at the same time. The duration of the run of the first runner to cross the finish line is less than the duration of the run of the other runner.
CLIF Definition:	(forall ((d1 duration) (d2 duration)) (iff (< d1 d2) (and (≤ d1 d2) (not (= d2 d1)))))
OCL Definition:	context duration def: "duration1 < duration2"(d2: duration) : Boolean = self."duration1 ≤ duration2"-->contains(d2) and not "duration1 = duration2"-->contains(d2)

8.2.2 Duration Operations

From a mathematical point of view, the extension of 'duration' is a vector space over the real numbers. That is, two operations – addition and scalar multiplication – are defined on durations. They operations obey the following axioms:

Axiom V.1 (Addition is Closed): If d1 and d2 are durations, then d1 + d2 is a duration.

Axiom V.2 (Addition is Associative): If d1, d2, d3 are durations, then (d1 + d2) + d3 = d1 + (d2 + d3).

Axiom V.3 (Addition is Commutative): If d1 and d2 are durations, then d1 + d2 = d2 + d1.

Axiom V.4 (Additive Identity): There is a duration D0 such that, for every duration d1, d1 + D0 = d1.

Axiom V.5 (Additive Inverse): For each duration d1, there is a duration d2, such that d1 + d2 = D0.

Note: The existence of the inverse (-d1) is a mathematical necessity for the vector space. Whether it has physical meaning is quite another thing entirely.

Axiom V.6 (Scalar multiplication is closed): if d1 is a duration and n1 is a number, n1 * d1 is a duration.

Axiom V.7 (Scalar multiplication is distributive over durations): if d1 and d2 are durations and n1 is a real number, n1 * (d1 + d2) = (n1 * d) + (n1 * d2)

Axiom V.8 (Scalar multiplication is distributive over reals): if d1 is a duration, and n1 and n2 are numbers, (n1 + n2) * d1 = n1 * d1 + n2 * d1.

Corollary: For all durations d1, 0 * d1 = D0

Corollary: If n1 is a number and d1 is a duration, then n1 * d1 = D0 iff n1 = 0 or d1 = D0

Corollary (Ratio): If d1 and d2 are durations and not d2 = D0, then there exists a number n1 such that d2 = n1 * d1.

We call n1 the "ratio of d2 to d1."

Note that the above does not depend on the concept 'time unit.' In fact, the usefulness of 'time unit' depends on this property.

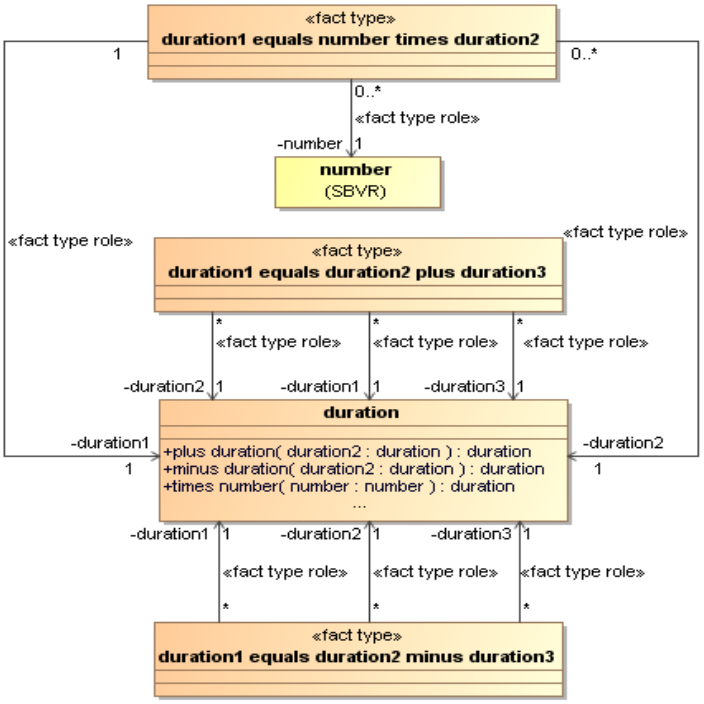

Figure 8.11 - Duration Operations

duration$_3$ *equals* duration$_1$ *plus* duration$_2$

Synonymous Form: duration$_3$ = duration$_1$ *plus* duration$_2$

Synonymous Form: duration$_1$ *plus* duration$_2$ *gives* duration$_3$

Synonymous Form: duration$_1$ + duration$_2$ *gives* duration$_3$

Synonymous Form: duration$_1$ *plus* duration$_2$

Synonymous Form: duration$_1$ + duration$_2$

Note: This is a "ground concept" that cannot be defined in terms of other concepts.

Example: Some race consists of a run and a swim. For each racer, the duration of the race is the duration of the run *plus* the duration of the swim.

Note: Axiom V.1 (Addition is closed) is incorporated into this verb concept by definition.

Axiom V.2 (Addition is Associative): If d1, d2, d3 are durations, then (d1 + d2) + d3 = d1 + (d2 + d3).

Necessity: If a duration$_4$ *equals* a duration$_1$ *plus* a duration$_2$, and a duration$_5$ *equals* duration$_4$ *plus* duration$_3$, and a duration$_6$ *equals* duration$_2$ *plus* duration$_3$, then duration$_5$ *equals* duration$_1$ *plus* duration$_6$.

CLIF Axiom:
```
(forall ((d1 duration) (d2 duration)
        (d3 duration) (d4 duration))
  (if (+ (+ d1 d2) d3)
      (+ d1 (+ d2 d3)))
```

Axiom V.3 (Addition is Commutative): If d1 and d2 are durations, then d1 + d2 = d2 + d1.

Necessity: Each duration$_1$ *plus* duration$_2$ *equals* duration$_2$ *plus* duration$_1$.

CLIF Axiom:
```
(forall ((d1 duration) (d2 duration))
  (= (+ d1 d2)
     (+ d2 d1)))
```

Axiom V.4 (Additive Identity): There is a duration D0 such that, for every duration d1, d1 + D0 = d1.

D0

Definition: duration that each duration$_1$ *equals* duration$_1$ *plus* D0

duration$_3$ *equals* duration$_1$ *minus* duration$_2$

Synonymous Form: duration$_3$ = duration$_1$ - duration$_2$

Synonymous Form: duration$_1$ *minus* duration$_2$ *gives* duration$_3$

Synonymous Form: duration$_1$ - duration$_2$ *gives* duration$_3$

Synonymous Form: duration$_1$ *minus* duration$_2$

Synonymous Form: duration$_1$ - duration$_2$

Definition: duration$_1$ *equals* duration$_3$ *plus* duration$_2$

Note: There are no time intervals with negative durations, but negative durations can arise when subtracting one duration from another duration. In common usage, a negative duration is a combination of a direction and a magnitude.

Example: A business process consists of task A immediately followed by task B. In any instance of the business process, the duration of task B is the duration of the entire business process minus the duration of task A.

Axiom V.5 (Additive Inverse): For each duration d1, there is a duration d2, such that d1 + d2 = D0.

Necessity: D0 *equals* each duration$_1$ *plus* some duration$_2$.

CLIF Axiom:
```
(forall ((d1 duration))
  (exists ((d2 duration))
   (= D0 (+ d1 d2)))))
```

duration$_2$ *equals* number *times* duration$_1$

Synonymous Form: duration$_2$ *equals* duration$_1$ *times* number

Synonymous Form: duration$_2$ = number * duration$_1$

Synonymous Form: duration$_2$ = duration$_1$ * number

Synonymous Form: number *times* duration$_1$

Synonymous Form: duration$_1$ *times* number

Synonymous Form: number * duration$_1$

Synonymous Form: duration$_1$ * number

Definition: duration$_2$ is the result of duration$_1$ *plus* duration$_1$, repeated number times

Example: 50 seconds *equals* 50 *times* 1 second

Axiom V.6 (Scalar multiplication is closed): if d1 is a duration and n1 is a number, n1 * d1 is a duration.

Note: This is implied by the SBVR and CLIF definition, hence no additional SBVR Necessity is required.

CLIF Axiom:
```
forall( ((n1 number) (d1 duration))
  (exists (d2 duration)
    (* n1 d1 d2)))
```

Axiom V.7 (Scalar multiplication is distributive over durations): if d1 and d2 are durations and n1 is a number, n1 * (d1 + d2) = (n1 * d1) + (n1 * d2)

Necessity: If a duration$_3$ *equals* a number$_1$ *times* (a duration$_1$ *plus* a duration$_2$) then duration$_3$ *equals* (number$_1$ *times* duration$_1$) *plus* (number$_1$ *times* duration$_2$).

CLIF Axiom:
```
(forall  ((d1 duration) (d2 duration) (n1 number))
  (exists ((d3 duration) (d4 duration) (d5 duration)
           (d6 duration) (d7 duration))
    (if (and
          (+ d1 d2 d3)
          (* n1 d3 d4)
          (* n1 d1 d5)
          (* n1 d2 d6)
          (+ d5 d6 d7))
      (= d4 d7))))
```

Axiom V.8 (Scalar multiplication is distributive over reals): if d1 is a duration, and n1 and n2 are numbers, (n1 + n2) * d1 = n1 * d1 + n2 * d1.

Necessity: If (a duration$_1$ *plus* a duration$_2$) *times* a number$_1$ *equals* a duration$_3$ then duration$_3$ *equals* (number$_1$ *times* duration$_1$) *plus* (number$_1$ *times* duration$_2$).

CLIF Axiom:
```
(forall  ((d1 duration) (n1 number) (n2 number))
   (= (* (+ n1 n2) d1)
      (+ (* n1 d1) (* n2 d1)
```

Corollary: For all durations d1, 0 * d1 = D0.

Necessity: D0 *equals* 0 *times* each $duration_1$.

CLIF Axiom:
```
(forall ((d1 duration))
  (* 0 d1 D0))
```

Corollary: If n1 is a number and d1 is a duration, then n1 * d1 = D0 iff n1 = 0 or d1 = D0.

Necessity: D0 *equals* a $number_1$ *times* a $duration_1$ if and only if $number_1$ *equals* 0 or $duration_1$ *equals* D0.

CLIF Axiom:
```
(forall (n1 number) (d1 duration))
  (exists ((d2 duration))
    (iff (and
           (* n1 d1 d2)
           (= d2 D0))
         (or
           (= n1 0)
           (= d1 D0)))))
```

Corollary (Ratio): If d1 and d2 are durations and not d2 = D0, then there exists a number n1 such that d2 = n1 * d1.

Necessity: If a $duration_1$ *does* not *equal* D0, then a $duration_2$ *equals* a $number_1$ *times* $duration_1$.

CLIF Axiom:
```
(forall ((d1 duration))
  (if (not (= d1 D0))
     (exists ((d2 duration) (n1 number))
        (* d1 n1 d2))))
```

8.2.3 Relationships between 'Duration' and 'Time Interval'

The intent of the 'duration' concept is to measure time intervals, but the model presented above is a mathematical abstraction that does not depend on time intervals for its properties. What makes it useful is the following set of relationships between durations and time intervals.

Each time interval has a unique duration attribute that is a measure of its size, i.e., the amount of time the time interval occupies. This attribute is mathematically a function that maps time intervals into durations. This mapping function is sometimes called the "range" of a time interval, and some times called the "measure" of a time interval. Following SBVR practice, this specification calls it the duration *of* a time interval.

This sub clause describes the only special cases in which the durations of constructed time intervals are well-defined.

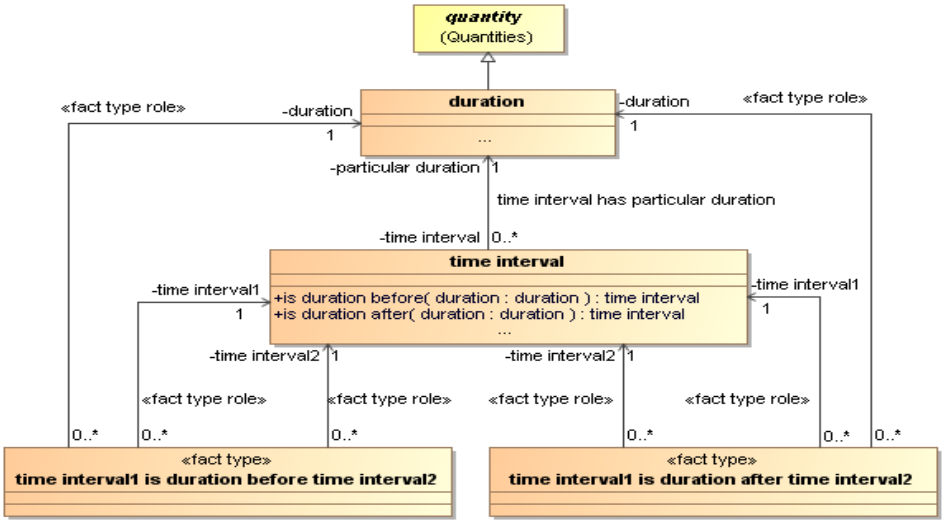

Figure 8.12 - Relationships between 'Duration' and 'Time Interval'

particular duration

Concept Type:	role
General Concept:	duration
Definition:	particular quantity *of* quantity kind 'duration'
Definition:	duration *of* some time interval
Reference Scheme:	a duration value that *quantifies* the particular duration
Example:	Particular duration of a particular meeting.

time interval *has* particular duration

Synonymous Form:	particular duration *of* time interval
Synonymous Form:	time interval *has* duration
Synonymous Form:	duration *of* time interval
Example:	The duration of Henry V's life is given by the duration value "35 years."

Axiom D.1: Each time interval has exactly one duration.

Necessity:	Each time interval *has* exactly one duration.
CLIF Axiom:	(forall ((t "time interval") (d1 duration) (d2 duration)) (if (and ("time interval has duration" t d1) ("time interval has duration" t d2)) (= d1 d2)))

Axiom D.2: Every time interval has a positive duration.

Necessity:	The duration *of* each time interval *is greater than* D0.
CLIF Axiom:	(forall ((t "time interval")) (> ("duration of" t) D0))

OCL Constraint:
```
context "time interval"
inv: self.duration > D0
```

Corollary: No time interval has duration D0.

Necessity: The duration *of* no time interval *equals* D0.

CLIF Axiom:
```
(forall ((t "time interval"))
  (not (= ("duration of" t) D0)))
```

OCL Constraint:
```
context "time interval"
inv: not self.duration = D0
```

Corollary: No time interval has a duration that is the additive inverse of the duration of any time interval. Thus, the vector space 'duration' is larger than the image of the time intervals.

Necessity: D0 *does* not *equal* D0 *minus* the duration *of* a time interval.

CLIF Axiom:
```
(forall ((t "time interval"))
  (not (= D0 (- (D0 t)))))
```

OCL Constraint:
```
context "time interval"
inv: not D0 = (D0 – self.duration)
```

Axiom D.3: If t1 and t2 are time intervals such that t1 is a *part of* t2, then D(t1) ≤ D(t2).

Necessity: If a time interval$_1$ *is a part of* a time interval$_2$, then the duration$_1$ *of* time interval$_1$ *is less than or equal to* the duration$_2$ *of* time interval$_2$.

CLIF Axiom:
```
(forall ((t1 "time interval") (t2 "time interval"))
  (if ("time interval1 is part of time interval2" t1 t2)
     (<= ("duration of time interval" t1)
        ("duration of time interval" t2)) )
```

OCL Constraint:
```
context "time interval"
inv: self."is part of"-->forAll(t2 |
   self.duration ≤ t2.duration)
```

Axiom D.4. If t1 and t2 are time intervals such that t1 meets t2, D(t1+t2) = D(t1) + D(t2).

Necessity: If a time interval$_1$ *meets* a time interval$_2$, then the duration$_3$ *of* time interval$_1$ *plus* time interval$_2$ *is equal to* the duration$_1$ *of* time interval$_1$ *plus* the duration$_2$ *of* time interval$_2$.

CLIF Axiom:
```
(forall ((t1 "time interval") (t2 "time interval"))
  (if ("time interval1 meets time interval2" t1 t2)
     (exists ((t3 "time interval") (d3 duration))
       (and
        ("time interval3 equals time interval1 plus
          time interval2" t3 t1 t2)
        ("duration3 equals duration1 plus duration2"
          d3 ("duration of" t1) ("duration of" t2))
        (= d3 ("duration of" t3))))))
```

OCL Constraint:
```
context "time interval"
inv: self.meets-->forAll(t2 |
    (self.duration + t2.duration) = (t1+t2).duration
```

Corollary: If t1 and t2 are time intervals such that t1 *starts* t2, then D(t1 *starts* t2 *complementing* t3) = D(t2) – D(t1).

Necessity:	If a time interval$_1$ *starts* a time interval$_2$ *complementing* a time interval$_3$, then the duration$_3$ *of* time interval$_3$ *is equal to* the duration$_2$ *of* time interval$_2$ *minus* the duration$_1$ *of* time interval$_1$.
CLIF Axiom:	(forall ((t1 "time interval") (t2 "time interval")) (exists ((t3 "time interval")) (if ("time interval1 starts time interval2 complementing time interval3" t1 t2 t3) (= ("duration of" t3) (- ("duration of" t2) ("duration of" t1))))))
OCL Constraint:	context "time interval" inv: self.starts-->forAll(t2 | self."time interval1 starts time interval2 complementing time interval3"(t2).duration = (t2.duration)- t1.duration

Corollary: If t1 and t2 are time intervals such that t1 *finishes* t2, then D(t1 *finishes* t2 *complementing* t3) = D(t2) – D(t1).

Necessity:	If a time interval$_1$ *finishes* a time interval$_2$ *complementing* a time interval$_3$, then the duration$_3$ *of* time interval$_3$ *is equal to* the duration$_2$ *of* time interval$_2$ *minus* the duration$_1$ *of* time interval$_1$.
Necessity:	(forall ((t1 "time interval") (t2 "time interval")) (exists ((t3 "time interval")) (if ("time interval1 finishes time interval2 complementing time interval3" t1 t2 t3) (= ("duration of" t3) (- ("duration of" t2) ("duration of" t1))))))
OCL Constraint:	context "time interval" inv: self.finishes-->forAll(t2 | self."time interval1 finishes time interval2 complementing time interval3"(t2).duration = (t2.duration)- t1.duration

time interval$_2$ *is* duration *before* time interval$_1$

Synonymous Form:	duration *before* time interval$_1$ *is* time interval$_2$
Synonymous Form:	duration *before* time interval$_1$
Synonymous Form:	time interval$_1$ - duration
Definition:	time interval$_2$ *is before* time interval$_1$ and time interval$_2$ *meets* some time interval$_3$ and time interval$_3$ *meets* time interval$_1$ and duration *is* the duration *of* time interval$_3$
Necessity:	Duration $\geq$ D0.
Example:	The time interval that *is indicated by* "10:55" is the duration that is *quantified by* "7 minutes" *before* the time interval that *is indicated by* "11:02".

time interval$_2$ *is* duration *after* time interval$_1$

Synonymous Form:	duration *after* time interval$_1$ *is* time interval$_2$
Synonymous Form:	duration *after* time interval$_1$
Synonymous Form:	time interval$_1$ + duration
Definition:	time interval$_2$ *is after* time interval$_1$ and time interval$_1$ *meets* some time interval$_3$ and time interval$_3$ *meets* time interval$_2$ and duration *is* the duration *of* time interval$_3$
Necessity:	Duration ≥ D0.
Example:	The time interval that *is indicated by* "1:49:53" is the duration that is *quantified by* "1 hour 49 minutes" *after* the time interval that *is indicated by* "00:00:53."

8.3 Temporal Concepts for Situations

This sub clause provides a vocabulary for relating situations to time intervals and durations; that is, it provides the basic vocabulary for writing rules or facts about the relationship between situations, events or activities and time. This treatment is motivated by the discussion in [Parsons] and [Menzel].

This specification relies on the idea of 'possible world' that is introduced in SBVR and derived from [Plantinga] – a specific collection of things and relationships that could be described by a set of consistent assertions (an SBVR 'fact model'), regardless of how that world relates to what we perceive as reality. Further, this specification uses the term 'universe of discourse' (or 'world of interest') to refer to the particular possible world that is chosen as the basis for determining what is 'true' or 'actual' with respect to a use of the ontology for reasoning and decision making. The conventional first-order logic treatment of time is: a different time is a different (possible) world. This specification treats time as an aspect of every possible world, so that any possible world can have a present, a past, and a future.

Consider the law of monogamy as it exists in some countries:

> It is prohibited that a person *is married to* more than one person.

This rule is correct only on the understanding that the rule is evaluated at a point in time, as specified in this document. A version of the rule that uses the concepts defined in this sub clause to make this understanding explicit is:

> If a person$_1$ *is married to* some person$_2$ *occurs for* some time interval, it is prohibited that person$_1$ *is married to* another person$_3$ *during* the time interval.

This specification uses four principal concepts:

1. occurrence: res that is an individual event, activity, situation, or circumstance that occurs in the universe of discourse and that *occurs for* exactly one time interval
2. situation model: res that is an abstract model or conceptualization of an event, activity, situation, or circumstance that may occur in some possible world. A situation model is said to be *exemplified by* its occurrences in a given world, and it is possible that there are none.
3. individual situation model: situation model that *has* at most one occurrence in each possible world
4. general situation model: situation model that *is* not an individual situation model

By presenting its own model, the Date-Time team hopes to describe what it believes it needs to relate situations to time. Different terms are used to avoid confusion between the models. The Date-Time team is eager to rationalize this model with SBVR and continues in discussion with the SBVR RTF towards that end.

The concepts defined here are parallel to the BPMN ideas of an activity/event model element (situation model) and an activity/event instance (occurrence).

The concept 'occurrence' is a foundational idea that is abstracted in the concept 'situation model.' Vocabularies and guidance statements normally refer to the abstraction, that is situation models, because they are intended to apply to all occurrences of the situation models. Facts may explicitly mention occurrences. For example, a contract may have a guidance statement about payments against the contract. The guidance statement would be about one or more situation models. The contract might also state facts, such as the date of signing of the contract, as an occurrence.

This clause describes how situation models and occurrences relate to time, via sub clauses that address:

1. Temporal behavior of occurrences
2. The relationship between situation models and time
3. The relationship between propositions and time
4. Schedules

Clause 8.3 defines concepts that relate states of affairs to tense and aspect as used in languages.

8.3.1 Situation models and occurrences

The principal concepts – situation model, occurrence, individual situation model, and general situation model – and the definitive relationships are depicted in Figure 8.13 and described below.

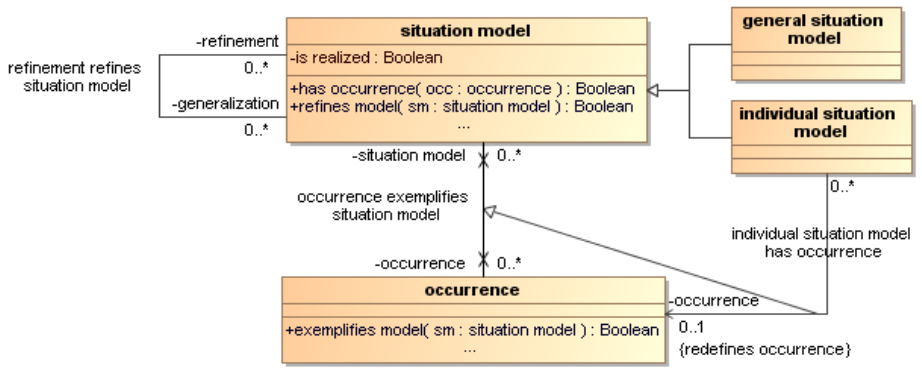

Figure 8.13 - Situation Models and Occurrences

situation model

Definition: res that is an abstract model or conceptualization of an event, activity, situation, or circumstance that may occur in some possible world

Necessity: Each situation model *is* either a general situation model or an individual situation model.

Example: "John is writing a book" is proposition that *describes* a situation model that is exemplified by each occurrence of John actually writing a book in the universe of discourse.

Note: A situation model is an abstract entity. It is not a concept and is not the actual event or situation. A situation model *is described by* a proposition (see 8.3.6).

occurrence

Definition: res that is an individual event, activity, situation, or circumstance that occurs in the universe of discourse and that *occurs for* exactly one time interval

Note: The occurrence is not a conceptualization of the situation or a model of the situation. It is an actual situation at some place and time in the possible world chosen for the universe of discourse. An occurrence of 'fire' can burn you.

Example: If a possible world includes all of December 2010, the physical flight of an aircraft from Washington to Minneapolis on December 20, 2010 from 7:00 to 9:00 EST is an occurrence in that world. In a possible world that is described by a fact model that includes flights, the flight of the aircraft exists. Any statement about all flights includes the particular flight. It *occurs within* December 2010 and *within* December 20, 2010, but it *occurs for* only the specified 2-hour time interval. It *occurs throughout* every time interval that is within that 2-hour time interval.

occurrence *exemplifies* situation model

Synonymous Form: situation model *has* occurrence

Definition: the occurrence is a realization of the situation model

Possibility: Each occurrence *exemplifies* zero or more situation models.

Possibility: Each situation model *has* zero or more occurrences.

Note: A situation model has an occurrence in a possible world only if the intersection between the occurrence interval of the occurrence (see Synonymous Form) and the time interval that is the window of time of the possible world is not empty. If the intersection is empty, the situation model does not have an occurrence in the possible world.

Example: The occurrence that *is described by* "EU-Rent rents car 123 to customer abc" *exemplifies* the situation model that *is described by* "EU-Rent rents a car to a customer."

individual situation model

Definition: situation model that *has* at most one occurrence in each possible world

Definition: Each individual situation model *has* at most one occurrence.

Example: The situation model that *is described by* the proposition "EU-Rent was incorporated on January 1, 2003" is an individual situation because it has just one occurrence.

general situation model

Definition: situation model that *is* not an individual situation model

Note: This concept is defined in contrast to 'individual situation model' not because there is any characteristic that distinguishes 'general situation model' from 'situation model'.

Note: A situation model is a general situation model if it can *be exemplified by* more than one occurrence in some possible world, even when it cannot have more than one occurrence in the possible world chosen to be the universe of discourse.

Possibility: Each general situation model *has* more than one occurrence.

Example: The situation model that *is described by* "EU-Rent rents a car to a customer" is a general situation model if and only if there are multiple occurrences described by this situation model.

refinement

Definition: situation model that *has* no occurrence that *does* not *exemplify* a given situation model

Concept Type: role

refinement *refines* situation model

Synonymous Form: situation model *has* refinement

Definition: Each occurrence *of* the refinement *exemplifies* the situation model

Example: The individual situation model *described by* "flight 123 from Washington to Minneapolis on December 20, 2010 arrives at 2pm" *refines* the general situation model *described by* "flight from Washington to Minneapolis arrives at 2pm."

Note: The *refines* fact type defines a partial ordering relationship among situation models that is analogous to the specialization/subtype relationship among concepts.

generalization

Definition: situation model that *is exemplified by* each occurrence *of* a given situation model

Concept Type: role

situation model *has* generalization

Definition: Each occurrence *of* the situation model *exemplifies* the generalization

Note: This is the inverse relationship to situation model *has* 'refinement'.

situation model *is realized*

Definition: There exists at least one occurrence that *exemplifies* the situation model in the universe of discourse (the chosen possible world)

Necessity: If situation model$_1$ *refines* situation model$_2$ and situation model$_2$ *is realized* then situation model$_1$ *is realized*.

8.3.2 Occurrences and Time

An occurrence is an actual happening in the world of interest. This sub clause provides a vocabulary for relating occurrences to time intervals and durations.

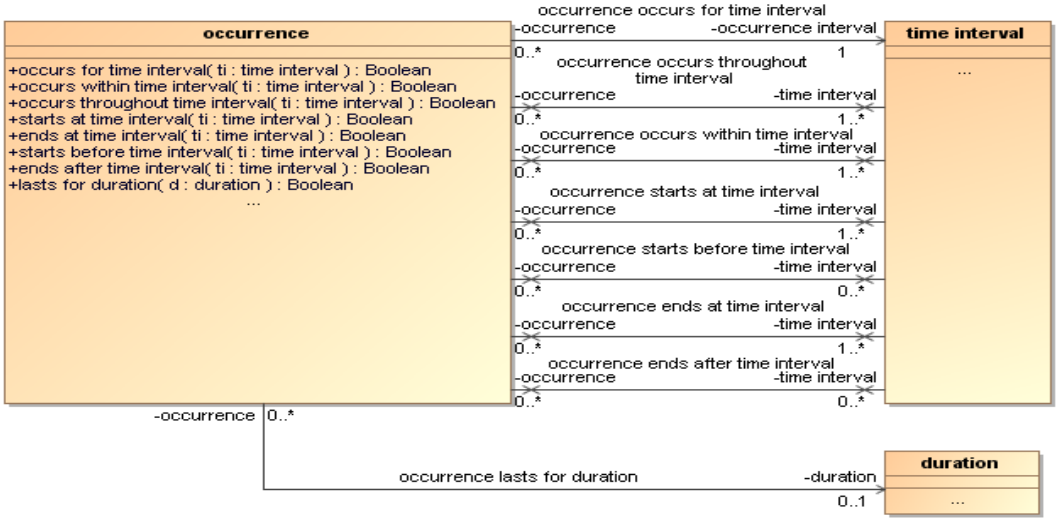

Figure 8.14 - Occurrences and Time

occurrence *occurs throughout* time interval

Synonymous Form: occurrence *throughout* time interval

Definition: the occurrence happens continuously, without interruption, in each time interval$_2$ that *is part of* the time interval

Note: This is a "primitive concept" – the fundamental relationship between occurrences and time. It cannot be defined in terms of other concepts. The idea is that an occurrence occurs at all times in some sufficiently small time interval.

Possibility: The occurrence may *occur throughout* some time interval$_2$ that *is* not *part of* the time interval.

Note: That is, the occurrence could occur throughout a longer time interval that includes other time intervals.

Example: The occurrence of "Barack Obama is President of the U.S." *occurred throughout* March, 2009.

occurrence *occurs within* time interval

Synonymous Form: occurrence *within* time interval

Synonymous Form: occurrence *in* time interval

Synonymous Form: occurrence *occurs at* time interval

Synonymous Form: occurrence *at* time interval

Synonymous Form: occurrence *during* time interval

Synonymous Form: time interval *covers* occurrence

Definition: the occurrence *occurs throughout* some time interval$_2$ that *is part of* the time interval

CLIF Definition:

```
(forall ((o "occurrence") (t "time interval"))
  (iff ("occurrence occurs within time interval"
        o t)
       (exists ((t2 "time interval"))
```

```
(and
 ("time interval1 is part of time interval2"
  t2 t)
 ("occurrence occurs throughout
   time interval" o t2)))))
```

OCL Definition:

```
context "occurrence"
def: "occurrence occurs within time interval"
    (t: "time interval") : Boolean
    t."part of"-->exists(t2 |
      self."occurrence occurs throughout
       time interval"(t2))
```

Example: The occurrence “William the Conqueror defeats Harold Godwineson in battle” *occurs within* the time interval that *has* the time coordinate “14 October 1066”.

occurrence interval

Concept Type: role

General Concept: time interval

Definition: the time interval that a given occurrence *occurs for*, i.e., the time span from the start of the occurrence to the end of the occurrence

occurrence *occurs for* occurrence interval

Synonymous Form: occurrence *occurs over* occurrence interval

Synonymous Form: occurrence *for* occurrence interval

Synonymous Form: occurrence *over* occurrence interval

Synonymous Form: occurrence *has* occurrence interval

General Concept: situation model *occurs for* time interval

Definition: the occurrence *occurs throughout* the occurrence interval and the occurrence *does not occur within* some time interval$_2$ that *meets* the occurrence interval and the occurrence *does not occur within* some time interval$_3$ that *is met by* the occurrence interval

CLIF Definition:

```
(forall (o "occurrence") (t "time interval"))
  (iff "occurrence occurs for time interval" o t)
    (and
      ("occurrence occurs throughout time interval"
       o t)
      (exists ((t2 "time interval"))
        (and ("time interval1 meets time interval2" t2 t)
            (not ("occurrence occurs within
                time interval" o t2))))
      (exists ((t3 "time interval"))
        (and ("time interval1 meets time interval2" t t3)
            (not ("occurrence occurs within
                time interval" o t3)))))))
```

OCL Definition:

```
context "occurrence"
def: "occurrence occurs for time interval"
    (t: "time interval") : Boolean
    self."occurrence occurs throughout time interval"
```

```
    (t)
  and self."is met by"-->forAll(t2 |
     not self."occurrence occurs throughout
           time interval"(t2))
  and self."meets"-->forAll(t3 |
     not self."occurrence occurs throughout
           time interval"(t3))
```

Note: The occurrence interval is the maximal time interval in which the individual occurrence occurs. The occurrence interval is immediately preceded and followed by time intervals when the occurrence does not happen.

Necessity: Each occurrence *occurs for* exactly one occurrence interval.

Possibility: Zero or more occurrences that *exemplify* a given general situation model *occur for* a given occurrence interval.

Example: The occurrence that is a specific flight of a specific aircraft *occurs for* the occurrence interval from the airplane's takeoff to the airplane's landing.

Note: No occurrence "recurs." An occurrence is an individual event; a "recurrence" is a different event, being distinguished by *occurring for* different time interval. What "recurs" is the common situation model.

Note: A former occurrence is an occurrence that *occurs over* some occurrence interval that *is in the past*. A planned occurrence is usually an occurrence that occurs over some future occurrence interval. A goal is a situation model that may have an occurrence at some future time.

Note: The occurrence interval is an essential intrinsic property of an occurrence, but it may not be known or specified, and it may not be relevant to every business model. For some uses, it may only be important that an occurrence happens *within* some time period, or that the situation model *occurs throughout* some time period.

occurrence *lasts for* duration

Synonymous Form: duration *of* occurrence

Definition: the occurrence *occurs for* some occurrence interval and the duration *is* the duration *of* the occurrence interval

CLIF Definition:

```
(forall ((o occurrence) (d duration))
  (iff ("occurrence lasts for duration" o d)
    (exists (t "time interval"))
      (and
        ("occurrence occurs for time interval"
          o t)
        ("time interval has duration" t d)))))
```

OCL Definition:

```
context "occurrence"
def: "occurrence lasts for duration"(d: duration)
    : Boolean
    self."occurrence occurs for time interval")
      .duration = d
```

Example: The duration of yesterday's meeting was 2 hours.

The following fact types are used primarily to enable us to talk about the beginning and end of occurrences in time.

occurrence *starts at* time interval

Definition: the time interval *starts* the occurrence interval *of* the occurrence or the occurrence interval *of* the occurrence *starts* the time interval or the occurrence interval *of* the occurrence *equals* the time interval

Note: '*Starts*' is the Allen relation (clause 8.1.3) between time intervals.

Note: The idea here is that the time intervals start together, but we know nothing about when they finish.

occurrence *starts before* time interval

Definition: the occurrence interval *of* the occurrence *precedes* the time interval or the occurrence interval *of* the occurrence *properly overlaps* the time interval

Note: '*Properly overlaps*' is the Allen relation (clause 8.1.3) between time intervals.

occurrence *ends at* time interval

Definition: the time interval *finishes* the occurrence interval *of* the occurrence or the occurrence interval *of* the occurrence *finishes* the time interval or the occurrence interval *of* the occurrence *equals* the time interval

Note: '*Finishes*' is the Allen relation (see 8.1.3) between time intervals.

Note: The idea here is that the time intervals finish together, but we know nothing about when they started. For example: "We should have a decision on the XYZ matter about the time that the contract review completes" means that the time interval at which the decision occurs will finish jointly with the contract review, irrespective of the times they started.

occurrence *ends after* time interval

Definition: the occurrence interval *of* the occurrence *follows* the time interval or the occurrence interval *of* the occurrence *is properly overlapped by* the time interval.

Note: 'Is properly overlapped by' is the Allen relation (clause 8.1.3) between time intervals

8.3.3 Temporal Ordering of Occurrences

Business processes and many rules constrain the time order of activities and events without specifying the actual times. And in general, these rules refer to activities and events as situation models. But only individual occurrences can occur in temporal order. So, in fact, only occurrences are ordered. The following verb concepts facilitate careful specification of such usages.

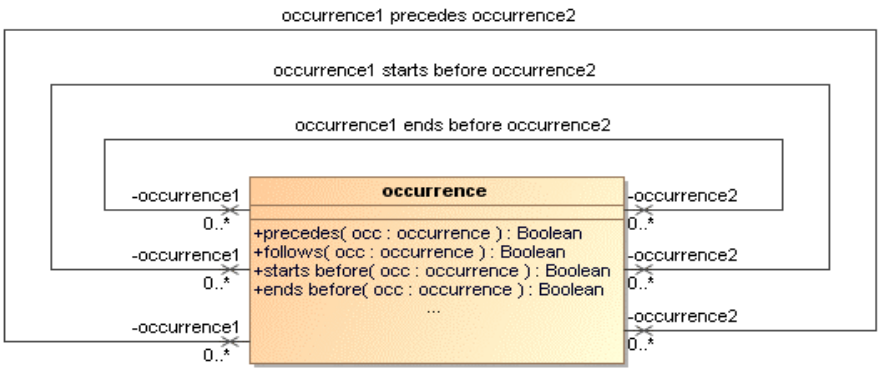

Figure 8.15 - Temporal Ordering of Occurrences

occurrence$_1$ *precedes* occurrence$_2$

Synonymous Form: occurrence$_2$ *follows* occurrence$_1$

Definition: the occurrence interval *of* occurrence$_1$ *precedes* the occurrence interval *of* occurrence$_2$

CLIF Definition:
```
(forall (o1 "occurrence") (o2 "occurrence))
  (iff ("occurrence1 precedes occurrence2"
        o1 o2)
       (forall ((t1 "time interval") (t2 "time interval"))
        (if
          (and
           ("situation model occurs for time interval"
              o1 t1)
           ("situation model occurs for time interval"
              o2 t2))
          ("time interval1 precedes time interval2"
           t1 t2))))
```

OCL Definition:
```
context "occurrence"
def: "occurrence1 precedes occurrence2"
    (o2: "occurrence") : Boolean
    self."occurs for" < o2."occurs for"
```

Necessity: If some occurrence$_1$ *precedes* some occurrence$_2$, and if the occurrence$_2$ *precedes* some occurrence$_3$, then occurrence$_1$ *precedes* occurrence$_3$.

CLIF Axiom:
```
(forall ((o1 "occurrence") (o2 "occurrence)
      (o3 "occurrence"))
  (if (and
       ("occurrence1 precedes occurrence2"
        o1 o2)
       ("occurrence1 precedes occurrence2"
        o2 o3))
      ("occurrence1 precedes occurrence2"
       o1 o3)))
```

OCL Constraint:
```
context "occurrence"
inv: self."precedes"-->exists(o2 |
      o2."precedes"-->exists(o3 | implies
       self."precedes"-->contains(o3)))
```

Note: This verb concept permits comparing the time order of two occurrences.

Example: On each airplane flight, the airplane takes off before the airplane lands.

occurrence$_1$ *starts before* occurrence$_2$

Synonymous Form: occurrence$_2$ *starts after* occurrence$_1$

Definition: the occurrence interval *of* occurrence$_1$ *starts before* the occurrence interval *of* occurrence$_2$

CLIF Definition:
```
(forall (o1 "occurrence") (o2 "occurrence))
  (iff ("occurrence1 starts before occurrence2"
        o1 o2)
       (forall ((t1 "time interval") (t2 "time interval"))
```

```
(if
  (and
   ("situation model occurs for time interval"
      o1 t1)
   ("situation model occurs for time interval"
      o2 t2))
  ("time interval1 starts before time interval2"
      t1 t2))))
```

OCL Definition:
```
context "occurrence"
def: "occurrence1 starts before occurrence2"
    (o2: "occurrence") : Boolean
  self."occurs for".
   "time interval starts before time interval"(o2."occurs for")
```

Note: This verb concept permits comparing the starting times of two occurrences.

Example: The procession must not start before the band plays.

occurrence$_1$ *ends before* occurrence$_2$

Synonymous Form: occurrence$_2$ *ends after* occurrence$_1$

Definition: the occurrence interval *of* occurrence$_1$ *ends before* the occurrence interval *of* occurrence$_2$

CLIF Definition:
```
(forall (o1 "occurrence") (o2 "occurrence"))
  (iff ("occurrence1 ends before occurrence2"
       o1 o2)
      (forall ((t1 "time interval") (t2 "time interval"))
       (if
         (and
          ("situation model occurs for time interval"
             o1 t1)
          ("situation model occurs for time interval"
             o2 t2))
         ("time interval1 ends before time interval2"
             t1 t2))))
```

OCL Definition:
```
context "occurrence"
def: "occurrence1 ends before occurrence2"
    (o2: "occurrence") : Boolean
  self."occurs for".
   "time interval ends before time interval"(o2."occurs for")
```

Note: This verb concept permits comparing the ending times of two occurrences.

Example: The delivery must be completed before the contract expires.

8.3.4 Situation models and time

This sub clause provides the basic vocabulary for writing rules or facts about the relationship between situation models and time.

Business processes and many rules constrain the timing of activities and events. In general, these rules refer to activities and events using situation models. A process specification assumes that what is being described is the sequencing of occurrences in an individual instance of the process. That is, the individual occurrences are described by the nature of the

happening (the situation model) and whatever information identifies the process instance. The fundamental notion here is that a situation model 'occurs' at any time it is exemplified by an actual occurrence in the world of interest, as discussed in 8.3.1.

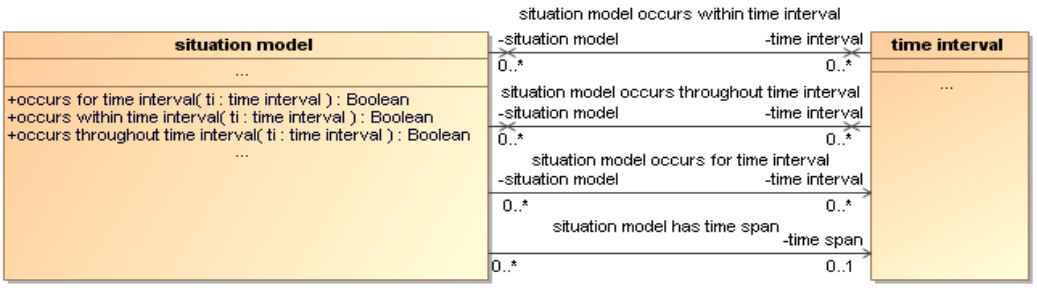

Figure 8.16 - Situation Models and Time

situation model *occurs throughout* time interval

Synonymous Form: situation model *throughout* time interval

Definition: some occurrence *of* the situation model *occurs throughout* the time interval

Possibility: A situation model may *occur throughout* no time interval.

situation model *occurs within* time interval

Synonymous Form: situation model *within* time interval

Synonymous Form: situation model *in* time interval

Synonymous Form: situation model *at* time interval

Synonymous Form: situation model *during* time interval

Definition: some occurrence *of* the situation model *occurs within* the time interval

Example: The situation model "soldiers are engaged in battle" *occurred within* the time interval that *has* the time coordinate "14 October 1066".

Example: "Flight 70 landed in Minneapolis at 9:12 on May 13, 2011."

situation model *occurs for* time interval

Definition: some occurrence *of* the situation model *occurs for* the time interval

Necessity: If the situation model *is* an individual situation model then the individual situation model *occurs for* at most one time interval.

Possibility: If the situation model *is* a general situation model then the general situation model *occurs for* more than one time intervals.

Note: For an individual situation model, the time interval is unique. For a general situation model, the model and the time interval may uniquely identify an occurrence.

time span

Definition: the smallest time interval that contains the occurrence intervals *of* all the occurrences *of* a given situation model.

Concept Type: role

situation model *has* time span

Definition: the occurrence interval *of* each occurrence *of* situation model *is part of* time span and no time interval that *is part of* time span *is before* the occurrence interval *of* each occurrence *of* situation model and no time interval that *is part of* time span *is after* the occurrence interval *of* each occurrence *of* situation model

Note: A general situation model may specify a constraint on the time interval of all of its occurrences, by stating the time span for the general situation model, or stating a constraint on it. Individual situation models that refine the general situation model each resolve the time down to a particular occurrence interval that must be within the time span.

Example: "The meetings will be weekly for the next three months" describes a general situation model whose time span is the specified time interval of the next three months. There can be a schedule of these meetings, giving the particular time for each meeting, which is an individual situation model.

Example: The time span of all the discount offers (a general situation model) is within July 2011. A particular discount (an individual situation model) offer occurs for July 13 from 2-3pm.

Example: The proposition "the meetings are scheduled for each Monday of July 2011" *describes* a general situation model whose time span *is within* the time interval "July 2011." If the individual meetings are held, then they *occur within* the Mondays of July 2011.

8.3.5 Temporal ordering of situation models

Business processes and many rules constrain the time order of activities and events without specifying the actual times. And in general, these rules refer to activities and events as situation models. Only individual occurrences actually have temporal ordering, but assigning such an ordering to the situation models themselves constrains the ordering of the actual occurrences. The following verb concepts facilitate careful specification of such usages.

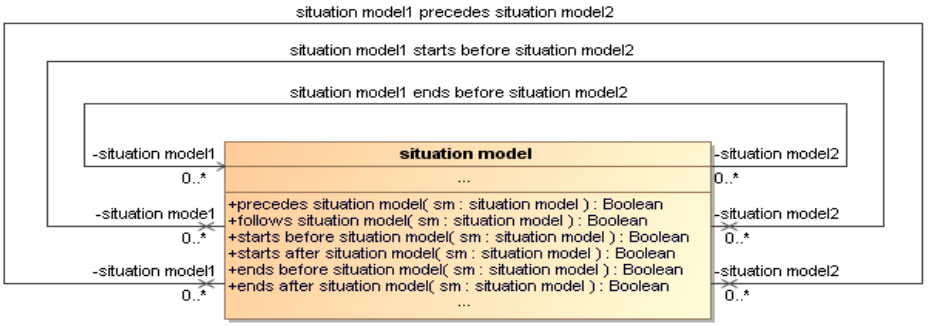

Figure 8.17 - Temporal Ordering of Situation Models

situation model$_1$ *precedes* situation model$_2$

Synonymous Form: situation model$_2$ *follows* situation model$_1$

Definition: each occurrence *of* situation model$_1$ *precedes* each occurrence *of* situation model$_2$

CLIF Definition:

```
(forall ((s1 "situation model") (s2 "situation model"))
  (iff ("situation model1 precedes situation model2"
        s1 s2)
       (forall ((o1 "occurrence") (o2 "occurrence"))
         (if
```

```
(and
 ("situation model has occurrence"
    s1 o1)
 ("situation model has occurrence"
    s2 o2))
("occurrence1 precedes occurrence2"
 o1 o2))))
```

OCL Definition:
```
context "situation model"
def: "situation model1 precedes situation model2"
    (s2: "situation model") : Boolean
    self."occurrence".precedes(s2."occurrence")
```

Note: This verb concept permits comparing the time order of two situation models. This is most useful in comparing individual situation models, but it has broader use.

Example: On each airplane flight, the airplane takes off before the airplane lands. (This compares two individual situation models.)

Example: The bank failures of the Great Depression (a general situation model) preceded World War II (an individual situation model).

situation model$_1$ *starts before* situation model$_2$

Synonymous Form: situation model$_2$ *starts after* situation model$_1$

Definition: each occurrence *of* situation model$_1$ *starts before* each occurrence *of* situation model$_2$

CLIF Definition:
```
(forall (s1 "situation model") (s2 "situation model))
  (iff ("situation model1 precedes situation model2"
        s1 s2)
       (forall ((o1 "occurrence") (o2 "occurrence"))
        (if
          (and
           ("situation model has occurrence"
              s1 o1)
           ("situation model has occurrence"
              s2 o2))
          ("occurrence1 starts before occurrence2"
           o1 o2))))
```

OCL Definition:
```
context "situation model"
def: "situation model1 precedes situation model2"
    (s2: "situation model") : Boolean
    self.occurrence."starts before"(s2.occurrence)
```

Note: This verb concept permits comparing the starting times of two situation models. This is primarily used for individual situation models.

Example: The procession must not start before the band plays.

situation model$_1$ *ends before* situation model$_2$

Synonymous Form: situation model$_2$ *ends after* situation model$_1$

Definition: each occurrence *of* situation model$_1$ *ends before* each occurrence *of* situation model$_2$

CLIF Definition:
```
(forall (s1 "situation model") (s2 "situation model))
  (iff ("situation model1 precedes situation model2"
        s1 s2)
```

```
(forall ((o1 "occurrence") (o2 "occurrence"))
  (if
    (and
     ("situation model has occurrence"
        s1 o1)
     ("situation model has occurrence"
        s2 o2))
    ("occurrence1 ends before occurrence2"
     o1 o2))))
```

OCL Definition:

```
context "situation model"
def: "situation model1 precedes situation model2"
    (s2: "situation model") : Boolean
    self.occurrence."ends before"(s2.occurrence)
```

Note: This verb concept permits comparing the ending times of two situation models. This is primarily used for individual situation models.

Example: The delivery must be completed before the contract expires.

8.3.6 Propositions, Situation Models, and Occurrences

In a static world that has no notion of change, there is a 1-to-1 relationship between propositions and states of the possible worlds: A proposition is true if the state it describes is the state of that world, and it is false if the state it describes is not the state of that world. (The SBVR model of states of affairs reflects this model.)

When temporal concepts are introduced into the formal logic model, a distinction must be made between two aspects of the SBVR concept 'proposition' – a 'meaning' that is either true or false, and that corresponds to at most one situation. This is because many propositions *describe* a single situation (a "general situation model") that may have multiple occurrences. For example, the proposition "each payment must precede delivery" is an SBVR way to state an obligation about the sequencing of payment and delivery, as might be given in a BPMN process model. In a given possible world, there may be many occurrences of payment and delivery, and thus many occurrences of payment preceding delivery.

In a temporal world, a logical sentence need not be true or false; it can be sometimes true and sometimes false, and therefore neither true nor false. That is, in a temporal world in which things change, a sentence that describes a state or event can describe a state that only occurs for some of the time, and perhaps occurs more than once at different times. The meaning of such a sentence is technically not a proposition, because it is not consistently true or false, but it still has occurrences. For example, the proposition "activity A precedes activity B for a given order" might be actually true of these activities for some orders and false for other orders.

Since SBVR does not make the distinction, we use 'proposition' here as the meaning of a sentence. In that sense, a proposition is the *description* of zero or more situations. A proposition *describes* a situation model and in so doing, it describes the occurrences, if any, that exemplify it.

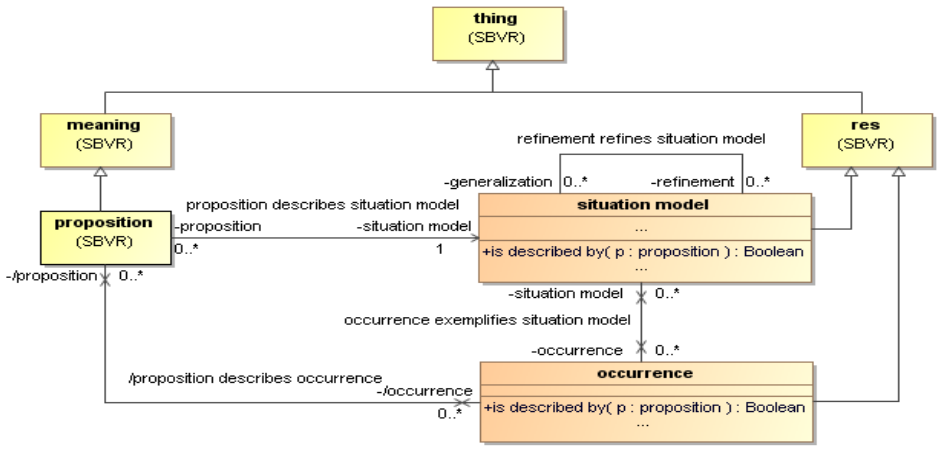

Figure 8.18 - Propositions, Situation Models, and Occurrences

proposition *describes* situation model

Synonymous Form: situation model *objectifies* proposition

Definition: the situation model models all actual occurrences that would make the proposition true, that is, the situation model is the interpretation of the proposition as a description of occurrences

Definition: The situation model *is* the bindable target *of* an objectification that *considers a* closed logical formulation that *means* the proposition.

Necessity: Each proposition *describes* exactly one situation model.

Necessity: Each situation model *is described by* zero or more propositions.

Example: The proposition “John is writing a book” is true during the time periods of each of the occurrences of John actually writing a book.

Example: “John was writing a book in 2009,” considered as a situation model, corresponds to at most one occurrence: the state in which John was writing a book during some time period within 2009. Thus it is described by the proposition “John was writing a book in 2009,” which is always either true or false. By comparison, “John is writing a book” represents a general situation model that may have had multiple occurrences in 2009 and in other years.

Note: Using the ‘proposition *describes* situation model’ fact type, the same proposition can be represented:

exists unitary p: proposition where
nominalize(p) "person dies"(Harold Godwineson)
exists unitary s: situation model where
"proposition describes situation model(p,s)
"situation model occurs within time interval"
(s, 14 October 1066)

proposition *describes* occurrence

Definition: In a possible world in which all time of interest overlaps the occurrence interval of the occurrence, the occurrence makes the proposition true.

Note: That is, the occurrence *exemplifies* the proposition in the sense of Plantinga (see [Menzel]).

Note: The occurrence is correctly described by each proposition that describes the situation model. In a possible world whose time frame is restricted to the occurrence interval of the occurrence, each proposition that describes the situation model *is true*.

Note: In a temporal world, the same proposition can describe several different occurrences, even when all the roles in the proposition are played by exactly the same things in all occurrences. What distinguishes the occurrences are the things that are not mentioned in the proposition. In particular, a proposition that does not mention time may describe different occurrences that have different occurrence intervals.

Example: Brazil wins the FIFA World Cup. That was true in 1994 and 2002, but false in 1992, 1998, 2006, and 2010. So the proposition "Brazil wins the FIFA World Cup" *describes* two occurrences in the last 20 years. It is true in the world of 1998, but it is false in the world of 2010, and it is both true and false, or neither, in a world that includes all of the last 20 years.

Possibility: A proposition *describes* zero or more occurrences (in a given possible world).

Possibility: An occurrence *is described by* zero or more propositions.

Note: Using the 'proposition *describes* occurrence' fact type, the 1066 proposition above can be formulated as:

```
exists unitary p: proposition where
      nominalize(p) "person dies"(Harold Godwineson)
   exists occ: occurrence where
      "proposition describes occurrence(p, occ)
   "occurrence occurs within time interval"
      (occ, 14 October 1066)
```

8.3.7 Schedules

Many situations recur according to a schedule. The nominal situation – the common description of all of them – is a situation model. This sub clause defines 'schedule' as a set of individual situation models of the same situation model.

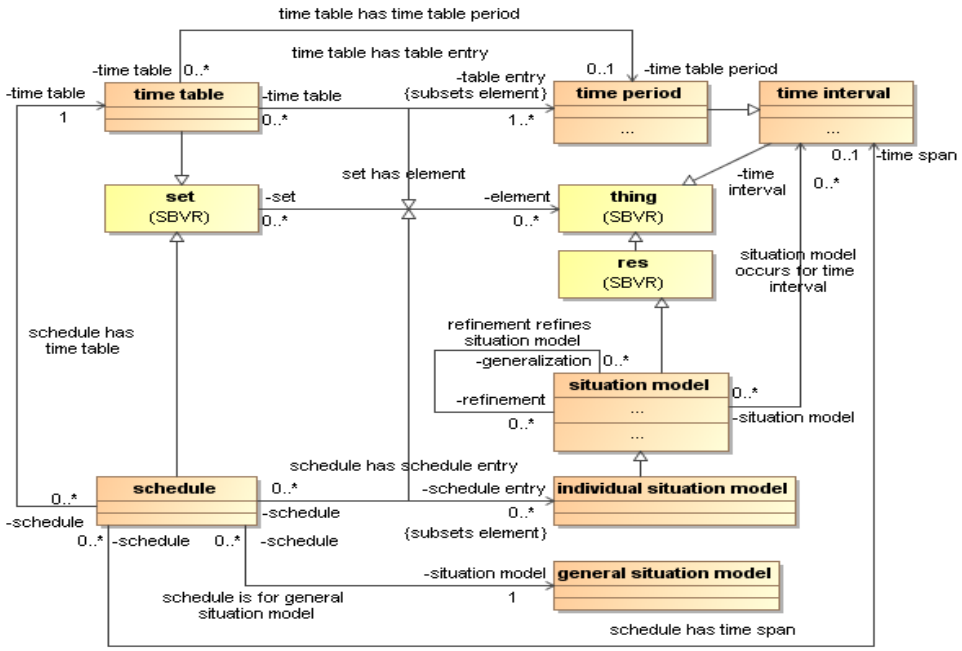

Figure 8.19 - Schedules

schedule

Definition: set *of* individual situation models that *has* a time table and that *is for* a general situation model and each individual situation model *refines* the general situation model and each individual situation model *occurs for* a table entry *of* the time table

CLIF Definition:

```
(forall s)
  (iff ("schedule" s)
    (and
      (exists ((g "general situation model")
               (tt "time table"))
        (and
          ("schedule is for general situation model" s g)
          ("schedule has time table" s tt)
          (set s)
          (forall ((i thing))
            (if
              ("set has element" s i)
              (and
                ("individual situation model" i)
.................("refinement refines situation model"
                    i g)
                (exists ((te "table entry"))
                  (and
                    ("time table has table entry" tt te)
                    ("situation model occurs for time
                      interval" i te)))
                        )))))))
```

Note: A hypothetical schedule may be defined in a fact model used for planning. The fact model models a possible world in which the individual situation models *occur for* planned time intervals.

Example: A class schedule describes a set of class meetings. Each scheduled meeting is an individual situation model. The meetings that actually happen are occurrences.

schedule *is for* general situation model

Synonymous Form: general situation model *according to* schedule

Synonymous Form: general situation model *has* schedule

Definition: each individual situation model *of* the schedule *refines* the general situation model

Necessity: A schedule *is for* exactly one general situation model.

Example: A rental car is serviced *according to* a maintenance schedule.

Example: Flights from NY to Dubai are *according to* a flight schedule. The general situation model is 'flights from NY to Dubai.'

Example: Flights according to a flight schedule. The same flight schedule might be for both general situation models "flights" and "flights from NY to Dubai."

schedule *has* time table

Synonym: time table *for* schedule

Definition: each table entry *of* the time table *is* the occurrence interval *of* some individual situation that *is* a member *of* the schedule

CLIF Definition:

```
(forall ((s "schedule") (tt "time table"))
  (iff ("schedule has time table" s tt)
    (forall (ti "time interval")
      (and
        ("time table has table entry" tt ti)
        (exists (i "individual occurrence"))
          (and
            ("set has member" s i)
            ("individual situation model occurs over time
              interval" i ti)))))))
```

OCL Definition:

```
context "schedule"
def: "schedule has time table"(t: "time table")
    : Boolean =
    t."table entry"-->forAll(te |
      self.member-->exists(i |
        i."individual situation model occurs over time
          interval"(te)))
```

Possibility: Some table entries *of* the schedule *are in the past*.

Possibility: Some table entries *of* the schedule *are in the future*.

time span

Concept Type: role

General Concept: time interval

Definition: the smallest time interval that *includes* the time intervals *of* all individual situation models in a given schedule

Note: The time span is the "convex hull" of a schedule.

schedule *has* time span

Synonymous Form: time span *of* schedule

Definition: the time table period *of* the time table *of* the schedule.

CLIF Definition:
```
(forall ((s "schedule") (span "time interval")))
  (iff ("schedule has time span" s span)
   (and
    (forall ((tt "time table") (ti "time interval"))
     (if
       (and ("schedule has time table" s tt)
            ("time table has time table period" tt ti))
       (= span ti)
```

OCL Definition:
```
context schedule
def: "schedule has time span"(t: time interval)
    : Boolean
    self."time table"."time table period" == t
```

Necessity: Each schedule *has* exactly one time span.

Necessity: The occurrence interval *of* each individual situation model *of* the schedule *is part of* the time span *of* the schedule.

9 Organizing Time (normative)

This Clause defines various concepts that impose a structure on time, in order to simplify designations for time intervals.

9.1 Time Units

As with other quantity kinds, durations are measured in terms of units. Unlike other quantity kinds, common time units are not simple ratios of each other. This makes for considerable complexity in specifying these time units. The details of this complexity are deferred to Clause 10.

The fundamental source of the complexity is that one of the main time units, 'year,' is incommensurable with other time units, such as 'month' and 'day.' This fact is due to the derivation of "year" and "day" from physical characteristics of our world.

9.1.1 Time Unit Concepts

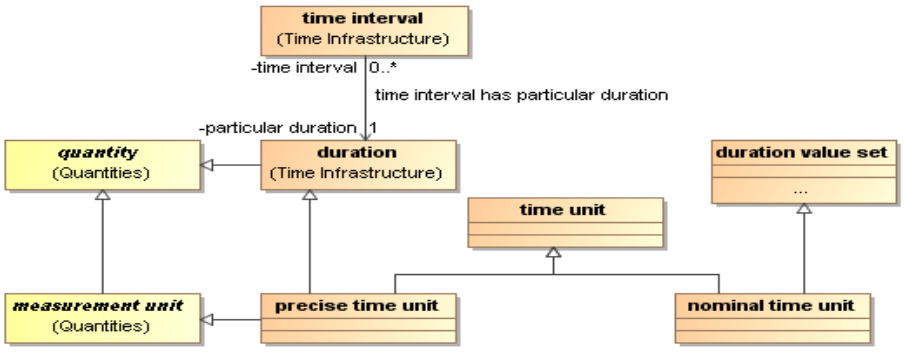

Figure 9.1 - Time Units

time unit

Definition: precise time unit or nominal time unit

Example: year, week, hour

precise time unit

Definition: measurement unit that *is* a duration

Note: [ISO 8601] defines "hour" and "minute" precisely. Leap seconds are added or deleted from a calendar day, not an hour of day or minute of hour, even though a leap second is designated as 23:59:60 of the day in which it is added. However, this document does not model leap seconds.

Example: second, minute, hour

nominal time unit

Definition: duration value set that is defined and adopted by convention, whose purpose is to express duration values that vary over the duration value set

Note: Each nominal time unit can be traced to counting cycles of some natural phenomenon. Historically the phenomena have been astronomical: the orbital cycles of the Earth and the Moon and the diurnal cycle of the Earth. Unfortunately for time keeping, these cycles are incommensurable, requiring intercalary time periods to maintain synchronization. Leap years have been used since 46 BC with the introduction of the Julian calendar to keep the calendar aligned with seasons of the year. The Gregorian calendar, introduced in 1582, refined the Julian calendar, which had an accumulated drift of 10 days at that time (over a time interval of 1628 years), which was corrected by skipping over the dates between October 5-15, 1582 [Inter Gravissimas].

Note: 'Year' and 'month' are said to be 'nominal time units' because of the effects of leap days.

Note: This specification ignores leap seconds because they are not relevant to most business affairs.

Example: Year defined as {365 days, 366 days}.

Example: Month defined as {28 days, 29 days, 30 days, 31 days}. Each month on the Gregorian calendar is a choice of 28, 29, 30, or 31 days.

9.1.2 Standard Time Units

This sub clause provides standard concepts about times of day, as found in [ISO 8601] and [SI], and generally accepted around the world.

second

Synonym: s

Synonym: sec

Definition: the precise time unit that is equal to the amount of time required for 9 192 631 770 periods of the radiation corresponding to the transition between the two hyperfine levels of the ground state of the caesium 133 atom

Dictionary Basis: The International System of Units (SI) 2.1.1.3 'Unit of time (second)'

Note: The duration of a second is a constant. In 1972, the second broke with astronomy and went to an atomic clock standard.

millisecond

Synonym: ms

Source: SI

General Concept: derived unit, precise time unit

Definition: .001 seconds

microsecond

Synonym: µs

General Concept: derived unit, precise time unit

Source: SI

Definition: 10^{-6} second

nanosecond

Synonym: ns

General Concept: derived unit, precise time unit

Source: SI

Definition: 10^{-9} second

picosecond

Synonym: ps

General Concept: derived unit, precise time unit

Source: SI

Definition: 10^{-12} second

minute

Synonym: min

General Concept: derived unit, precise time unit

Source: ISO 31-1

Definition: the precise time unit that *is quantified by* '60 seconds'

hour

Synonym: h

General Concept: derived unit, precise time unit

Source: ISO 31-1

Definition: the precise time unit that *is quantified by* '3600 seconds'

day

Synonym: d

Definition: the precise time unit that *is* the duration *of* a calendar day

Definition: the precise time unit that *is quantified by* 86 400 seconds

Note: For business purposes, this document treats 'day' as a precise time unit, even though the duration of a calendar day varies due to leap seconds and due to discontinuities when a locality switches between standard time and daylight time. Domain vocabularies may define a nominal time unit that takes such variations into account for scientific, engineering, or academic purposes.

Note: In UTC, a leap second may be added or subtracted about every 18 months, on June 30 or December 31, as determined by the International Earth Rotation and Reference Systems Service (IERS). (There have been no subtractions to date, as the rotation of the Earth is slowing.) Leap seconds are used to keep UTC aligned with ephemeris time to within one second, and can only be predicted about six months in advance, because of irregularities in the Earth's rotation.

Note: Some time standards do not use leap seconds, notably GPS time and International Atomic Time (TAI), on which UTC is based.

Note: Different calendars may define "day" differently. Particularly, in calendars based on solar time rather than ephemeris time, the calendar day may be defined by sunrise to sunrise, sunset to sunset or noon to noon. In such cases, the duration of a calendar day varies cyclically through the calendar year by as much as half an hour, a phenomenon known as the Equation of Time. Solar time is measured by observations and instruments such as sun dials, ephemeris time is measured by clocks.

year

Definition: the nominal time unit that *is* the duration *of* some calendar year

Definition: the nominal time unit that *is quantified by* {365 days, 366 days}

Note: There are several methods for reckoning a year. The main method is the return of the Vernal Equinox. This is called a tropical year, whose length is 365.2424 days of 86 400 seconds. There are several other year schemes, whose length in days of 86 400 seconds varies from about 347 days to about 384 days, depending how a year is measured.

Note: The definition of a year is dependent on the use of a specific calendar. See "Gregorian year."

month

Definition: the nominal time unit that *is* the duration *of* some calendar month

Definition: the nominal time unit that *is quantified by* {28 days, 29 days, 30 days, 31 days}

Note: A lunar month is about 28 days, and is incommensurable with a year. Different calendars define the number of days in a month differently. And the same calendar may define different calendar months to have different numbers of days. The Gregorian calendar has 12 calendar months that were rather arbitrarily set to a certain number of days by Roman politicians, without synchronizing with the lunar cycle.

week

Source: ISO 8601 (2.2.9, 'week')

Definition: the precise time unit that *is* the duration *of* a calendar week

Definition: the precise time unit that *is quantified by* '604 800 seconds'

9.2 Time Scales

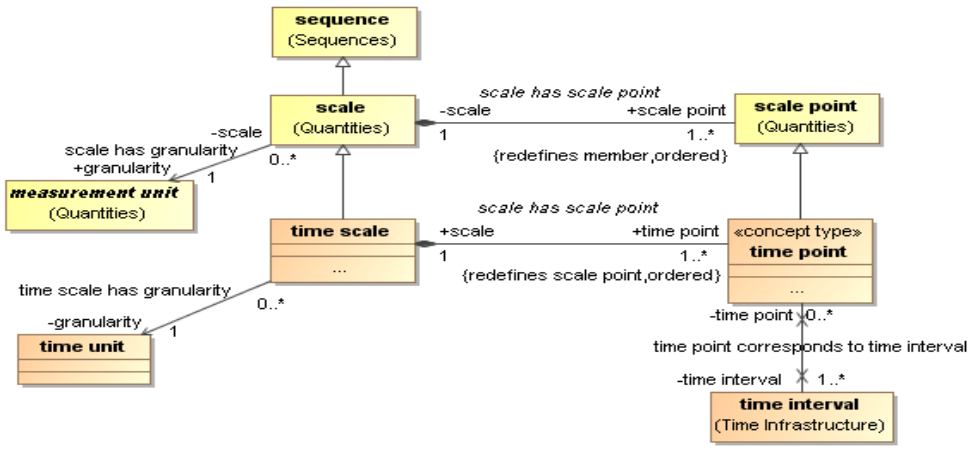

Figure 9.2 - Time Scales

time scale

Definition: scale that *has* a granularity that *is* a time unit and that each member *of* the scale *is* a time point

Necessity:	If a member *of* a time scale *has* a previous member then each time interval that *is* an instance *of* the member *is met by* some time interval that *is* an instance *of* the previous member.
Necessity:	If a member *of* a time scale *has* a next member then each time interval that *is* an instance *of* the member *meets* some time interval that *is* an instance *of* the next member.
Note:	These Necessities are really part of the definition of 'time scale'.
Dictionary Basis:	IEC 60050-111, ("time scale")
Dictionary Basis:	IEC 8601, (2.1.4, "time scale")
Definition:	system of ordered marks that can be associated with time intervals on the Time Axis, with one time interval being chosen as the reference point
Note:	[from ISO 8601] A time scale may among others be chosen as: • continuous, e.g., international atomic time (TAI) (see IEC 60050-713, item 713-05-18); • continuous with discontinuities, e.g., Coordinated Universal Time (UTC) due to leap seconds, standard time due to summer time and winter time; • successive steps, e.g., usual calendars, where the Time Axis is split up into a succession of consecutive time intervals and the same mark is attributed to all instants of each time interval; • discrete, e.g., in digital techniques.
Note:	[from ISO 8601] For physical and technical applications, a time scale with quantitative marks is preferred, based on a chosen initial instant together with a unit of measurement.
Note:	[from ISO 8601] Customary time scales use various units of measurement in combination, such as second, minute, hour, or various time intervals of the calendar such as calendar day, calendar month and calendar year.
Note:	[from ISO 8601] A time scale has a reference point which attributes one of the marks of the time scale to one of the instants, thus determining the attribution of marks to instants for the Time Scale.
Note:	Each semantic community should agree on a closed set of time scales.
Example:	The clock face of a traditional clock is a time scale.

granularity

Synonym:	resolution
Concept Type:	role
General Concept:	time unit
Dictionary Basis:	VIM (4.15, 'resolution (2)')
Definition:	the smallest duration that can be distinguished with a given time scale
Example:	"Second" as the granularity for a time scale in which each time point has the duration "1 second."

time scale *has* granularity

Definition:	The granularity *of* the time scale *is* the duration *of* the time points *of* the time scale.

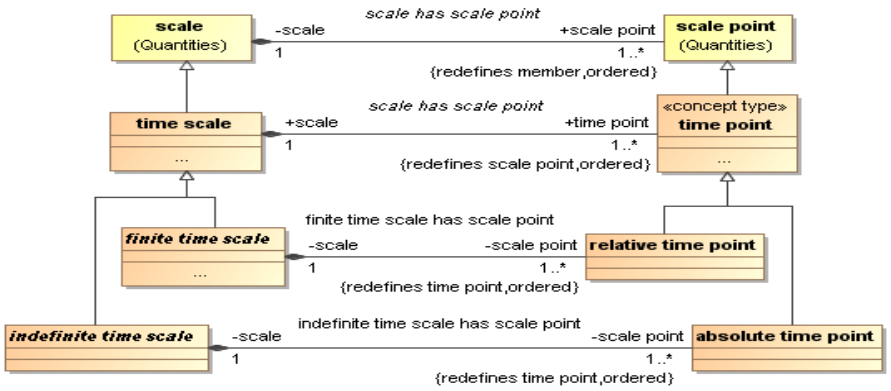

Figure 9.3 - Time Scale Kinds

finite time scale

Definition: time scale that *is* a finite scale

Note: A finite time scale is a time scale that has a cardinality, a first member, and a last member.

Example: the Gregorian year of months scale

Example: the hour of minutes scale

indefinite time scale

Definition: time scale that *is* an indefinite scale

Note: An indefinite time scale has no cardinality because it has no first member, no last member, or both.

Example: the Gregorian years scale

absolute time point

Definition: time point that *is of* an indefinite time scale

Necessity: Each absolute time point *corresponds to* exactly one time interval.

Example: The absolute time coordinate 'September 11, 2011' *indicates* an absolute time point.

relative time point

Definition: time point that *is of* a finite time scale

Necessity: Each relative time point *corresponds to* more than one time interval.

Example: The relative time coordinate 'September 11' *indicates* multiple relative time points, one for each calendar month.

9.3 Time Points

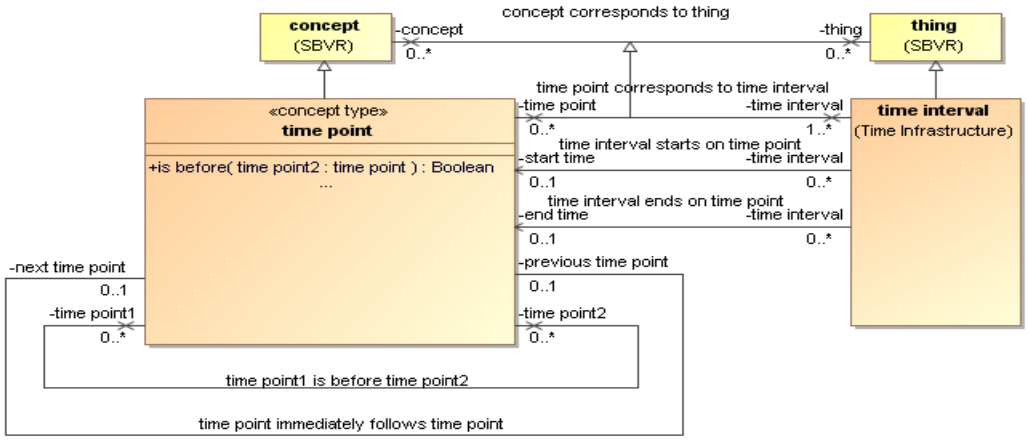

Figure 9.4 - Time Points

time point

Concept Type:	concept type
General Concept:	time period
Definition:	scale point that *is in* a time scale and that *specializes* the concept 'time interval'
Necessity:	The duration *of* each time interval that *is* an instance *of* the time point *is* the granularity *of* the time scale *of* the time point.
Note:	Each time point is a concept whose instances are time intervals.
Reference Scheme:	an occurrence *at* the time point
Reference Scheme:	a time coordinate that *indicates* the time point
Note:	This is a total reference scheme: every time point *is indicated by* at least one time coordinate, and some time points may *be indicated by* multiple time coordinates.
Example:	The Battle of Hastings *was on* “14 October 1066”. (This gives the Julian date of the battle at a granularity of “day”. If desired, the battle could be given more precisely as a time period within that calendar day.)

time interval *starts on* time point

Synonymous Form:	time point *starts* time interval
Definition:	some time interval that *is* an instance *of* the time point *starts* the time interval

time interval *ends on* time point

Synonymous Form:	time point *ends* time interval
Definition:	some time interval that *is* an instance *of* the time point *finishes* the time interval

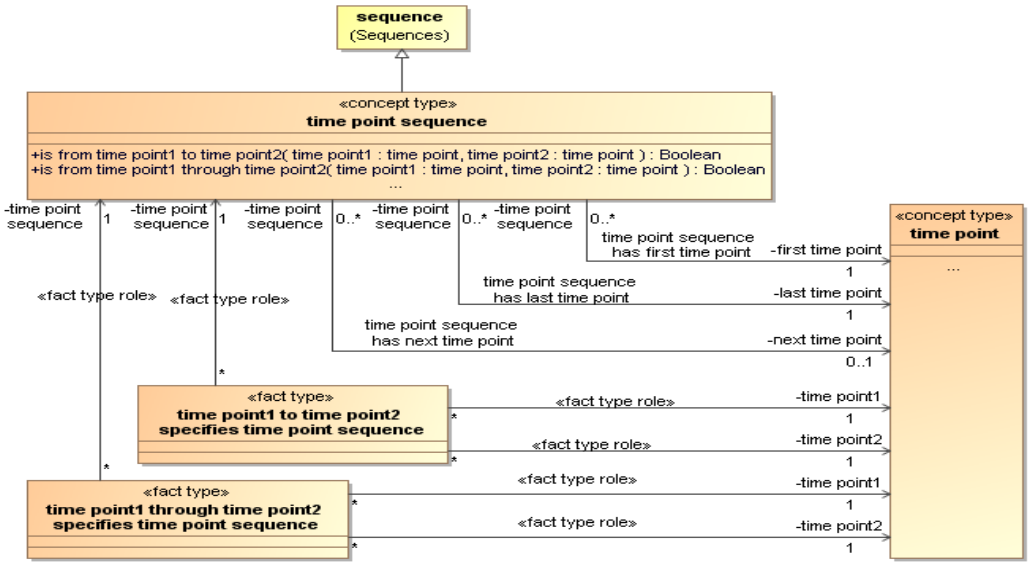

Figure 9.5 - Time Point Sequence

time point sequence

Definition:	consecutive sequence *of* time points *of* exactly one time scale
Reference Scheme:	The first time point *of* the time point sequence and the last time point *of* the time point sequence.
Reference Scheme:	The first time point *of* the time point sequence and the duration *of* the time point sequence.
Reference Scheme:	The last time point *of* the time point sequence and the duration *of* the time point sequence.
Reference Scheme:	The first time point *of* the time point sequence, if the time point sequence *has* no last time point.
Reference Scheme:	The last time point *of* the time point sequence, if the time point sequence *has* no first time point.
Necessity:	Each time point sequence *includes* at least one time point.
Necessity:	Each time point sequence *has* at most one first time point.
Necessity:	Each time point sequence *has* at most one last time point.
Note:	A time point sequence may be “open” at either or both ends; i.e., have no first time point or no last time point, or neither. (See forever.)
Example:	The time point sequence from July 1, 2009 to August 3, 2010.

time point sequence *corresponds to* time interval

Synonymous Form:	time interval *instantiates* time point sequence
Definition:	the time interval *starts on* the first time point *of* the time point sequence and *ends on* the last time point *of* the time point sequence and *includes* exactly one time interval that *is* an instance *of* the first time point *in* the time point sequence

time point sequence *has* duration

Definition: the duration *equals* the cardinality *of* the time point sequence *times* the granularity *of* the time point sequence

Example: The duration of the time point sequence consisting of Monday, Tuesday, and Wednesday is 3 days.

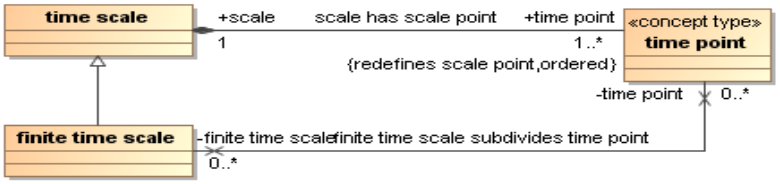

Figure 9.6 -Time Scale Subdivision

finite time scale *subdivides* time point

Definition: the finite time scale divides the time point into a number *of* time intervals that each *have* a duration, and the number *is* the cardinality *of* the finite time scale, and the duration *is* the granularity *of* the finite time scale

Necessity: The day of hours scale *subdivides* the Gregorian date 3 January 2010 into 24 time intervals, each of duration "hour"

9.4 Time Periods

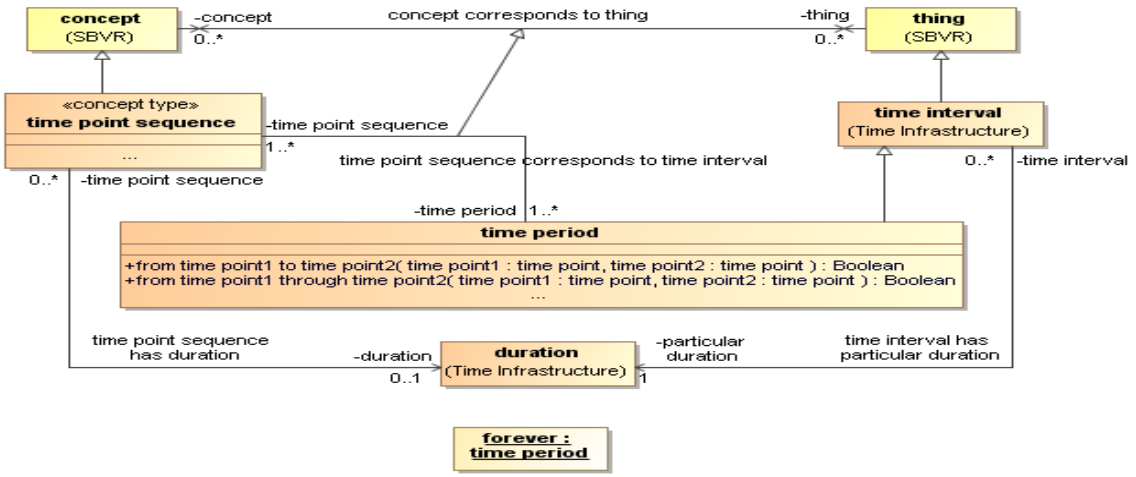

Figure 9.7 - Time Periods

time period

Definition: time interval that *instantiates* some time point sequence

first time point

Synonym: start time point

Concept Type: role

General Concept: time point

time point sequence *has* first time point

Synonymous Form: first time point *of* time point sequence

Definition: the first time point *of* the time point sequence is the first member *of* the time point sequence

Note: "... before July 1" refers to a time point sequence with no first time point.

Example: The time coordinate of the first time point of the time point sequence from July 1, 2009 to August 3, 2010 is July 1, 2009.

last time point

Synonym: end time point

Concept Type: role

General Concept: time point

time point sequence *has* last time point

Synonymous Form: last time point *of* time point sequence

Definition: the last time point *of* the time point sequence is the last member *of* the time point sequence

Note: "... after July 1" refers to a time point sequence with no last time point.

Example: The time coordinate of the last time point of the time point sequence from July 1, 2009 to August 3, 2010 is August 3, 2010.

time point$_1$ *through* time point$_2$ *specifies* time period

Synonymous Form: time point$_1$ *through* time point$_2$

Definition: time point$_1$ *is* the first time point *of* a time point sequence and time point$_2$ *is* the last time point *of* the time point sequence and the time point sequence *corresponds to* the time period

Necessity: The time scale *of* time point$_1$ *is* the time scale *of* time point$_2$.

Necessity: If the time scale *of* time point$_1$ *is* an indefinite time scale then time point$_1$ *through* time point$_2$ *specifies* exactly one time period.

Possibility: If the time scale *of* time point$_1$ is a finite time scale then time point$_1$ *through* time point$_2$ *specifies* more than one time period.

Note: Contrast '*through*' with '*to*.' '*Through*' is inclusive of time point$_2$, while '*to*' is exclusive of time point$_2$.

Example: "January *through* March", meaning the time interval of 3 months duration that starts with January and ends with March.

time point$_1$ *to* time point$_2$ *specifies* time period

Synonymous Form: time point$_1$ *to* time point$_2$

Definition: time point$_1$ *is* the first time point *of* a time point sequence and some time point$_3$ *is* the last time point *of* the time point sequence and time point$_2$ *is just before* time point$_3$ *in* the time point sequence and time point$_1$ *through* time point$_2$ *specifies* the time period

Necessity: The time scale *of* time point$_1$ *is* the time scale *of* time point$_2$.

Necessity: If the time scale *of* time point$_1$ *is* an indefinite time scale then time point$_1$ *through* time point$_2$ *specifies* exactly one time period.

Possibility: If the time scale *of* time point$_1$ is a finite time scale then time point$_1$ *through* time point$_2$ *specifies* more than one time period.

Note: Contrast '*through*' with '*to*.' '*Through*' is inclusive of time point$_2$, while '*to*' is exclusive of time point$_2$.

Example: "January *to* March", meaning the time interval of 2 months duration that starts with January and ends with February.

forever

General Concept: time period

Definition: the time period that *does* not *start on* some time point and *does* not *end on* some time point

9.5 Calendars

Calendars use time scales to impose structure on time.

9.5.1 Calendar Fundamentals

This sub clause contains definitions true of calendars in general.

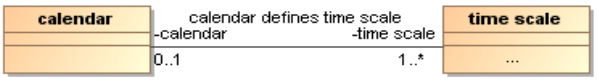

Figure 9.8 - Calendars

calendar

Definition: conceptual schema that *defines* one or more time scales

Note: This concept of calendar can include any date-time conceptual schema, of any granularity. This is more general than the usual calendar concept, which limits the finest granularity to "day". The two most prominent calendars are the Gregorian, whose finest granularity is "day", and the Universal Coordinated Time (UTC), whose finest granularity is "second". UTC uses the Gregorian calendar to get to a day and extends it to define the time of day down to a second calendar.

Note: There are many different calendars, some standard, some cultural, some defined for particular business needs.

Example: Gregorian calendar, lunar calendars, fiscal calendars, manufacturing calendars, tax calendars, religious calendars.

Reference Scheme: the time scales that *are defined by* a calendar

calendar *defines* time scale

Synonymous Form: time scale *of* calendar

Synonymous Form: time scale *on* calendar

Definition: the calendar specifies the details of the time scale

Example: The Gregorian calendar *defines* the Gregorian year time scale with other time scales.

9.5.2 Time of Day Time Scales, Time Points, and Time Periods

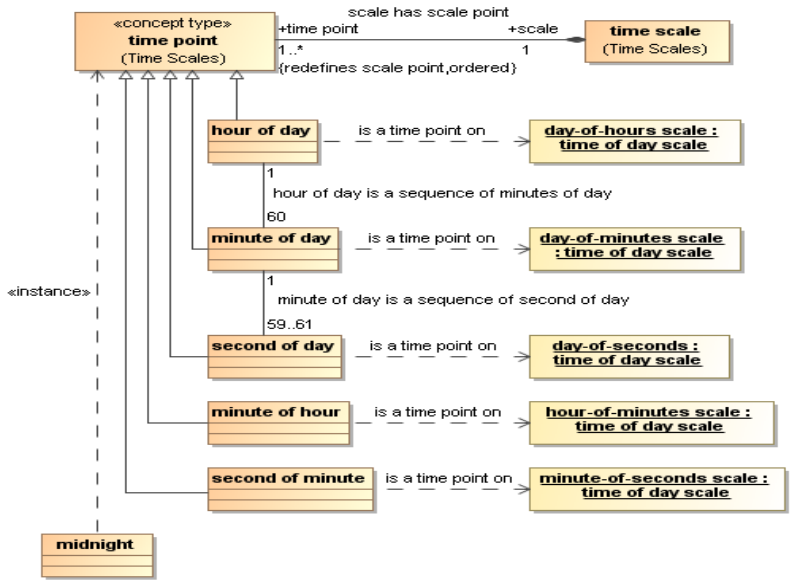

Figure 9.9 - Time of Day Time Scales, Time Points, and Time Periods

day of hours scale

Definition: the finite time scale *of* 24 'hour of day'

Necessity: The granularity *of* the day of hours scale *is* 'hour'.

Necessity: The index origin value *of* the day of hours scale *equals* 0.

Necessity: The first position *of* the day of hours scale *is* the index origin element *of* the day of hours scale.

day of minutes scale

Definition: the finite time scale *of* 1 440 'minute of day'

Necessity: The granularity *of* the day of minutes scale *is* 'minute'.

Necessity: The index origin value *of* the day of minutes scale *equals* 0.

Necessity: The first position *of* the day of minutes scale *is* the index origin element *of* the day of minutes scale.

day of seconds scale

Definition: the finite time scale *of* 86 400 'second of day'

Necessity: The granularity *of* the day of seconds scale *is* 'second'.

Necessity: The index origin value *of* the day of seconds scale *equals* 0.

Necessity: The first position *of* the day of seconds scale *is* the index origin element *of* the day of seconds scale.

hour of minutes scale

Definition: the finite time scale that *subdivides* 'hour of day' *into* 60 *of* 'minute of hour'

Necessity: The granularity *of* the hour of minutes scale *is* 'minute'.

Necessity: The index origin value *of* the hour of minutes scale *equals* 0.

Necessity: The first position *of* the hour of minutes scale *is* the index origin element *of* the hour of minutes scale.

minute of seconds scale

Definition: the finite time scale that *subdivides* 'minute of hour' *into* 60 *of* 'second of minute'

Necessity: The granularity *of* the minute of seconds scale *is* 'second'.

Necessity: The index origin value *of* the minute of seconds scale *equals* 0.

Necessity: The first position *of* the minute of seconds scale *is* the index origin element *of* the minute of seconds scale.

Example:

midnight

Definition: time point 00:00:00 within a calendar day

hour of day

Dictionary Basis: ISO 8601 (3.2.3)

Definition: time point that *is on* the day of hours scale and that is identified by the number of elapsed full hours since midnight on a given calendar day

Note: The standard that the hour of day is counted since midnight was established by the International Meridian Conference of 1884 [International Meridian].

Note: This is a relative time point because it is on a finite time scale.

minute of day

Definition: time point that *is on* the day of minutes scale and that is identified by the number of elapsed full minutes since midnight on a given calendar day

Note: This is a relative time point because it is on a finite time scale.

Example: "03:15" is the minute-of-day that has index 195

second of day

Definition: time point that *is on* the day of seconds scale and that is identified by the number of elapsed full seconds since midnight on a given calendar day

Note: This is a relative time point because it is on a finite time scale.

Example: "03:15:48" is the second-of-day that has index 11 748

minute of hour

Dictionary Basis: ISO 8601 (3.2.3)

Definition: time point that *is on* the hour of minutes scale and that is identified by the number of elapsed full minutes since the last full hour

Definition: This is a relative time point because it is on a finite time scale.

second of minute

Dictionary Basis: ISO 8601 (3.2.3)

Definition:	time point that *is on* the minute of seconds scale and that is identified by the number of elapsed full seconds since the last full minute
Definition:	This is a relative time point because it is on a finite time scale.
Note:	Business Calendar Concepts

9.5.3 Calendar Time Points and Time Periods

This sub clause defines categories of time points and time periods that *indicate* time intervals with duration 'day', 'month', or 'year', but are independent of any particular calendar design. These concepts are intended to apply to religious and cultural calendars as well as the Gregorian calendar.

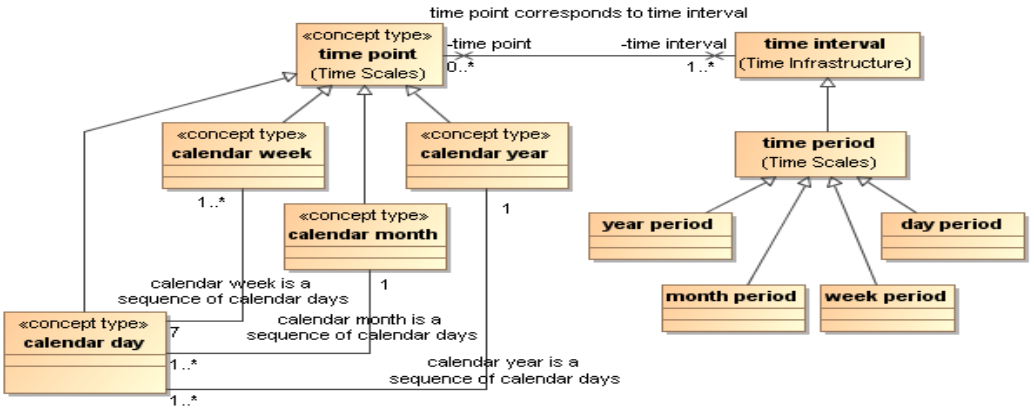

Figure 9.10 - Calendar Time Points and Time Periods

calendar year

Dictionary Basis:	ISO 8601 (2.2.13, 'calendar year')
Definition:	time point that is defined by a given calendar as a consecutive sequence of calendar days, during which approximately one orbital rotation of the Earth around the Sun is completed
Note:	See "Gregorian year".
Example:	the year 2008 (as defined by the Gregorian calendar)
Example:	the 15th year of the reign of the Pharaoh Akhenaton

calendar month

Definition:	time point that is defined by a given calendar as a consecutive sequence of calendar days in a calendar year, during which approximately one rotation of the Moon in its orbit around the Earth is completed
Example:	August, 1945 (as defined by the Gregorian calendar)
Example:	Ramadan in the 63rd year of the Prophet Mohammed

calendar day

Definition:	time point that is defined by a given calendar, and during which approximately one revolution of the Earth occurs on its axis
Example:	July 4, 1776 (as defined by the Gregorian calendar)

Example: The time period from sunrise in Rome on the Ides of March in the year 753 after the founding of the City to the following sunrise.

year period

Dictionary Basis: ISO 8601 (2.2.14, note 1)

Definition: time period which starts at a certain time of day at a certain calendar date of the calendar year and ends at the same time of day at the same calendar date of the next calendar year, if it exists. In other cases, the ending calendar date is defined by agreement.

Note: Calendar year is a period that starts and ends as defined by a calendar. Year period starts and ends at any time within a calendar year.

Example: The concept “fiscal year” defined as the year period from midnight of July 1 of one calendar year to midnight of July 1 of the following calendar year.

month period

Source: ISO 8601 (2.2.12, note 1)

Definition: time period that starts at a certain time of day at a certain calendar date of the calendar month and ends at the same time of day at the same calendar date of the next calendar month, if it exists. In other cases, the ending calendar date is defined by agreement.

Note: Calendar month is a period that starts and ends as defined by a calendar. Month period starts and ends at any time within a calendar month.

Example: From July 15 at noon to August 15 at noon.

day period

Definition: time period that begins and ends at the same local time of day on consecutive calendar days

Note: Calendar day is a period that starts and ends as defined by a calendar. Day period starts and ends at any time within a calendar day.

Example: Noon one calendar day to noon the following calendar day.

hour period

Definition: time period that begins and ends at the same minute of hour on consecutive hours of day

Example: 1:05 *to* 2:05

9.5.4 Time Zones

This sub clause provides the infrastructure for time zones.

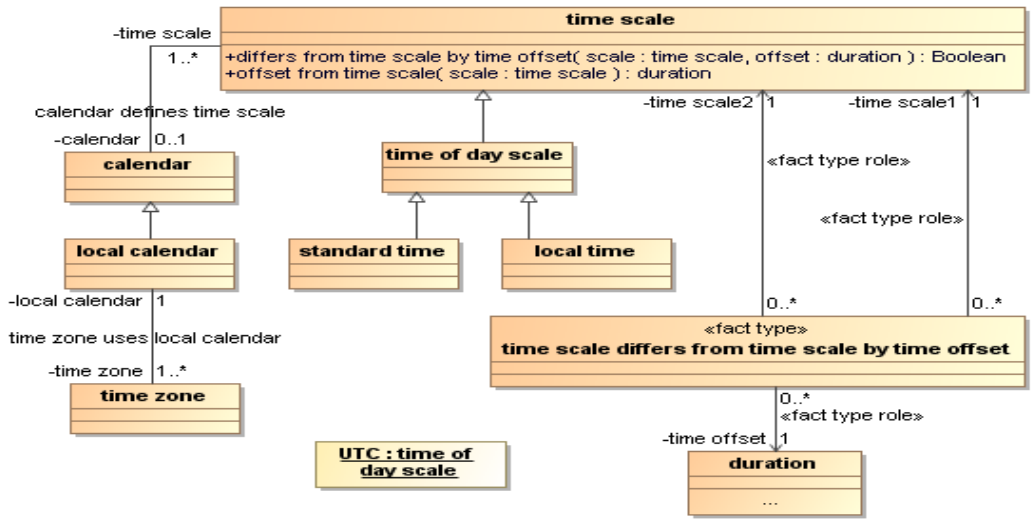

Figure 9.11 - Time Zones

UTC

Synonym:	Coordinated Universal Time
Source:	International Bureau of Weights and Mesaures (BIPM)
Source:	ISO 8601 (2.1.12)
Source:	IEC 60050-713
Definition:	calendar which forms the basis for the coordinated dissemination of standard frequencies and time signals; it corresponds exactly in rate with International Atomic Time, but differs from it by an integral number of seconds
Definition:	calendar that combines the Gregorian Calendar, a day-of-hours time scale, an hour-of-minutes time scale, and a minute-of-seconds time scale
Note:	UTC has granularity of second. Other calendars may be defined as multiples or fractions of seconds.
Note:	For the purposes of this vocabulary, all calendars are presumed to be correlated to UTC.
Note:	UTC is officially maintained by the BIPM in cooperation with national metrology institutes or observatories around the world. See http://www.bipm.org/en/scientific/tai/time_server.html.
Note:	The UTC scale is adjusted by the insertion of leap seconds (about every 18 months) to ensure approximate agreement with the time derived from the rotation of the Earth to within one second. This specification ignores leap seconds; day is considered to be a precise time unit of 86 400 seconds, not a nominal time unit of {86 400 days, 86 401 days}. Applications that entail leap seconds need to extend this specification to include them.

time offset

Concept Type:	role
Definition:	duration that characterizes a comparison of two calendars
Example:	Duration between a given indication (e.g., 12:00:00.000) on a clock set to local time and the same indication on a clock set to UTC time, where both of the clocks change at the same rate. Conventionally, a time offset is prefixed + to indicate that the local clock indication occurs before (is ahead of) the UTC indication, and – to indicate the local clock indication occurs after

(is behind) the UTC indication: local time – UTC time. The sign is styled as a keyword prefix to the duration to convey this precedence information. The number of a duration is always non-negative.

Example: Indian Standard Time – UTC = +5½ hours.

UTC time

Synonym: UTC of day

Dictionary Basis: ISO 8601 (2.1.13)

Definition: time point within a calendar day in accordance with UTC

time scale$_1$ *differs from* time scale$_2$ *by* time offset

Synonymous Form: time scale$_1$ – time scale$_2$ = time offset

Synonymous Form: time scale$_1$ = time scale$_2$ time offset

Definition: The time offset *is* the duration *between* a time point *on* time scale$_1$ and a time point *on* time scale$_2$.

Necessity: The time points *are identified by* the same time coordinate.

Necessity: The time scales *have* the same granularity.

Example: Pacific Standard Time *differs from* Eastern Standard Time *by* –3 hours. PST – EST = –3 hours. Note the minus sign on the right hand side is styled as a keyword. This means that a clock reading in EST occurs before the same clock reading in PST. Other way around, use +. See time offset.

Example: PST = EST – 3 hours. This form is commonly used to define time zones, where time scale$_2$ is UTC:
IST = UTC+5½ hours. Sometimes hours is implied and the "hours" symbol is omitted in time zone designations.

local calendar

Definition: calendar that *is* UTC except the day-of-hours time scale *differs from* the UTC day-of-hours time scale *by* a given time offset

Reference Scheme: a time offset *by* which the local calendar's day-of-hours time scale *differs from* the day-of-hours time scale of UTC

Example: Pacific Daylight Time (UTC–7 hours) , Eastern Standard Time (UTC–5 hours), British Summer Time (UTC+1 hour), Indian Standard Time (UTC+5½ hours)

Note: Many, but not all, local calendars are named. Calendar names are not unique, e.g., EST in the US and Australia. Many named local calendars may have the same time offset. For example, both Central European Standard Time and Algeria Standard Time are UTC+1 hour. A local calendar does not need to be named; it is identified by its time offset from UTC.

Note: ISO 8601 abbreviates time offsets by using only a signed four-digit number representing hours and minutes, omitting the "UTC" and "hours". Thus, IST is "+0530".

time zone

Definition: locale in which one or two local calendars is used

Note: When there are two calendars for a time zone, one is standard time and the other is daylight savings time. The dates and time of day for changing between them is determined by local authorities for each time zone.

time zone *uses* local calendar

Necessity: A given time zone *uses* exactly one local calendar *at a* given time.

standard time

Source: ISO 8601 (2.1.14)

Source: IEC 60050-111

Definition: time scale derived from Coordinated Universal Time, UTC, by a time shift established in a given location by the competent authority

local time

Synonym: local time of day

Source: ISO 8601 (2.1.16)

Definition: locally applicable time of day such as standard time of day, or a non-UTC base time of day

9.5.5 Gregorian Calendar

The Gregorian calendar is standardized in ISO 8601, and is widely used in business and everyday activities.

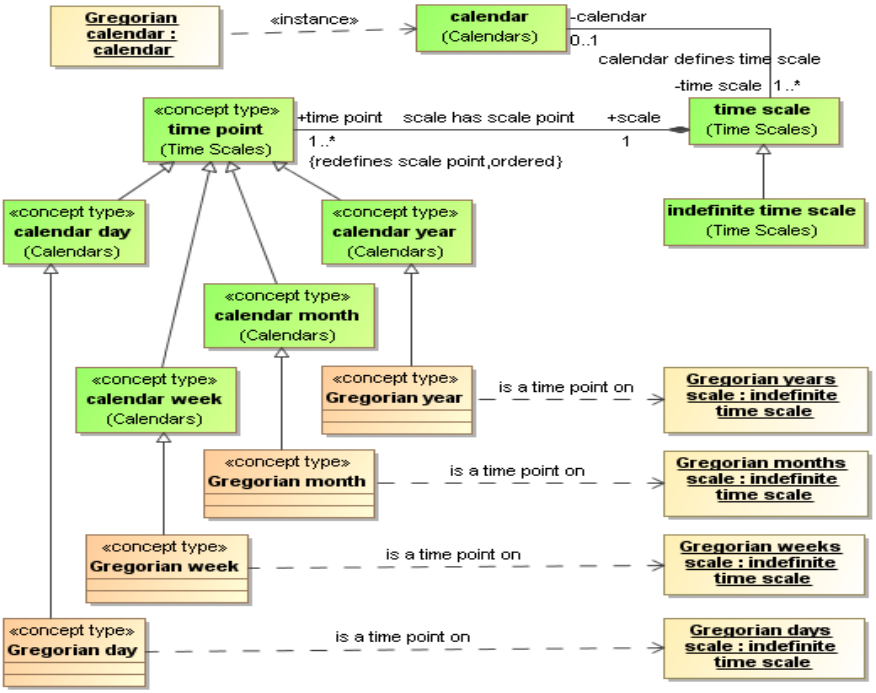

Figure 9.12 - Gregorian Indefinite Time Scales and Time Points

Gregorian calendar

Source: ISO 8601 (2.2.15, 'Gregorian calendar')

Definition: calendar in general use, introduced in 1582 to define a calendar year that more closely approximated the tropical year than the Julian calendar

Convention du Mètre

Definition:	time interval of the signing of the Convention du Mètre
Necessity:	The Convention du Mètre *is at* 20 May 1875.
Definition:	[ISO 8601] establishes the date of the signing of the Convention du Mètre, 20 May 1875, as the reference date for the Gregorian calendar.

Gregorian years scale

Definition:	the indefinite time scale *of* granularity 'year' and *of* 'Gregorian year' time points
Necessity:	The index origin value *of* the Gregorian years scale *equals* 1875.
Necessity:	The Convention du Mètre *is at* the index origin element *of* the Gregorian years scale.
Note:	This definition applies to the Gregorian calendar as recognized at the Prime Meridian at Greenwich in England during Standard Time. Other Gregorian years scales may be obtained by adding or subtracting time offsets, as discussed in sub clause 12.5.

Gregorian months scale

Definition:	the indefinite time scale *of* granularity 'month' and *of* 'Gregorian month' time points
Necessity:	The index origin value *of* the Gregorian months scale *equals* 22 493.
Note:	22 493 is (12 * 1875 - 1) + 5 (for the month of May) -1 (adjusting for index origin 1)
Necessity:	The Convention du Mètre *is at* the index origin element *of* the Gregorian years scale.
Note:	This definition applies to the Gregorian calendar as recognized at the Prime Meridian at Greenwich in England during Standard Time. Other Gregorian months scales may be obtained by adding or subtracting time offsets, as discussed in clause 12.5.

Gregorian days scale

Definition:	the indefinite time scale *of* granularity 'day' and *of* 'Gregorian day' time points
Necessity:	The index origin value *of* the Gregorian days scale *equals* 684 609.
Note:	The calendar reform instituted by Pope Gregory XIII [Inter Gravissimas] deleted 10 days from the Gregorian calendar, starting on 5 October 1582. The previous calendar day had index 577 739 on the Julian calendar, computed as 1581 years of 365 days plus 395 leap days + 279 days from 1 January 1852 to 5 October 1582. The following day was 15 October 1582. From that date to the Convention du Mètre on 20 May 1875, there were 106 870 calendar days including leap days. Therefore, the Convention happened on calendar day 684 609 of the Gregorian days scale.
Necessity:	The Convention du Mètre *is at* the index origin element *of* the Gregorian years scale.
Note:	This definition applies to the Gregorian calendar as recognized at the Prime Meridian at Greenwich in England during Standard Time. Other Gregorian days scales may be obtained by adding or subtracting time offsets, as discussed in sub clause 12.5.

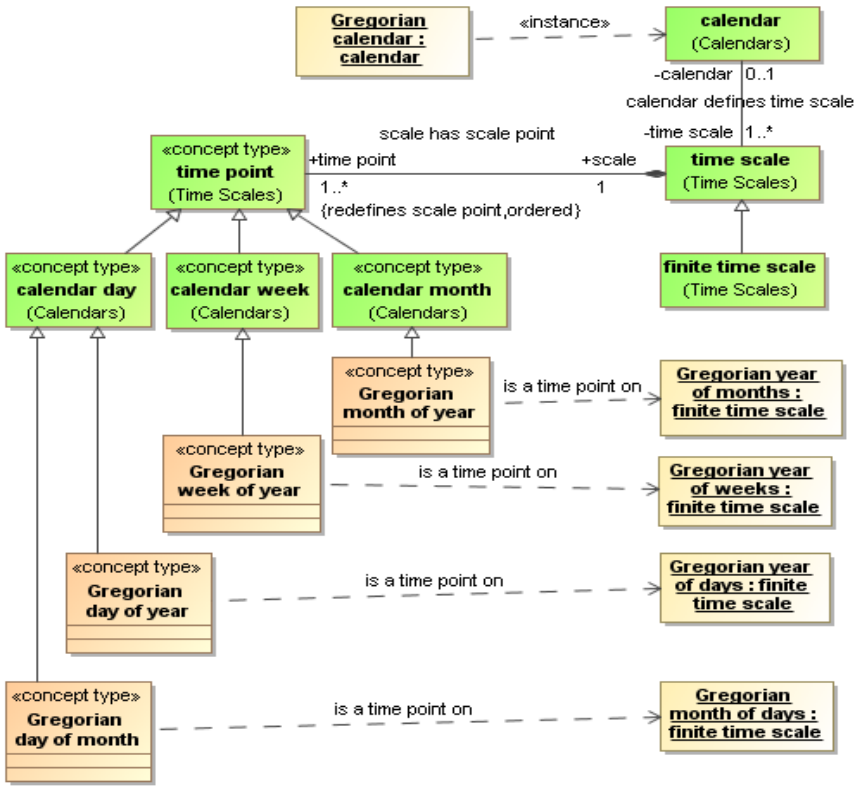

Figure 9.13 - Gregorian Finite Time Scales and Time Points

Gregorian year of months scale

Definition:	the finite time scale that *subdivides* 'Gregorian year' *into* 12 'Gregorian month of year'
Necessity:	The granularity *of* the Gregorian year of months scale *is* 'month'.
Necessity:	The index origin value *of* the Gregorian year of months scale *equals* 1.
Necessity:	The first element *of* the Gregorian year of months scale *is* the index origin element *of* the Gregorian year of months scale.

Gregorian year of days scale

Definition:	the finite time scale that *subdivides* 'Gregorian year' *into* 366 *of* 'Gregorian day of year'
Necessity:	The granularity *of* the Gregorian year of days scale *is* 'day'.
Necessity:	The index origin value *of* the Gregorian year of days scale *equals* 1.
Necessity:	The first element *of* the Gregorian year of days scale *is* the index origin element *of* the Gregorian year of days scale.
Note:	This time scale has 366 Gregorian days of year in order to accommodate leap years.

Gregorian month of days scale

Definition:	the finite time scale that *subdivides* 'Gregorian month' *into* 31 *of* 'Gregorian day of month'
Necessity:	The granularity *of* the Gregorian month of days scale *is* 'day'.

Necessity: The index origin value *of* the Gregorian month of days scale *equals* 1.

Necessity: The first element *of* the Gregorian month of days scale *is* the index origin element *of* the Gregorian month of days scale.

Note: This time scale has 31 Gregorian days of month in order to accommodate the longest Gregorian month.

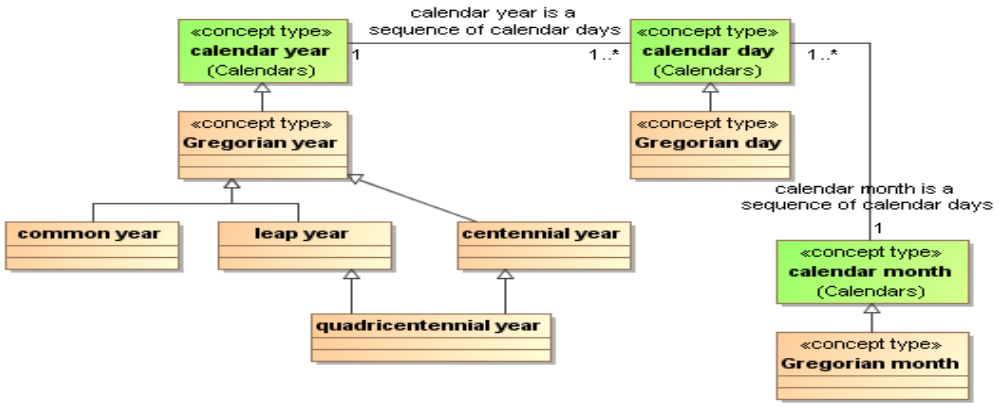

Figure 9.14 - Gregorian Time Points

common year

Definition: calendar year that *is on* the Gregorian year scale and the number *of* the calendar year, when divided by 4, generates a remainder that is not zero, or that *is* a centennial year

Note: This is an absolute time point because it is on an indefinite time scale.

leap year

Definition: calendar year that *is on* the Gregorian year scale and the number *of* the calendar year, when divided by 4, generates a remainder that is zero, and that *is* not a centennial year

Note: The rules for leap years were established by Pope Gregory XIII in [Inter Gravissimas]. These rules were eventually adopted by various civil governments and incorporated into [ISO 8601].

Note: This is an absolute time point because it is on an indefinite time scale.

centennial year

Source: ISO 8601 (2.2.18, 'centennial year')

Definition: calendar year that *is on* the Gregorian year scale that *is* not a quadricentennial year, and the number *of* the calendar year, when divided by 100, generates a remainder that is zero

Note: This is an absolute time point because it is on an indefinite time scale.

quadricentennial year

Source: ISO 8601 (2.2.18, 'centennial year')

Definition: calendar year that *is on* the Gregorian year scale and the number *of* the calendar year, when divided by 400, generates a remainder that is zero

Note: This is an absolute time point because it is on an indefinite time scale.

Gregorian year

Definition: common year or leap year that *is on* the Gregorian years scale

Note: This is an absolute time point because it is on an indefinite time scale.

Gregorian month

Definition: calendar month that *is on* the Gregorian months scale

Note: This is an absolute time point because it is on an indefinite time scale.

Gregorian month of year

Definition: calendar month that *is on* the Gregorian year of months scale

Note: This is a relative time point because it is on a finite time scale.

Gregorian day

Definition: calendar day that *is on* the Gregorian days scale

Note: This is an absolute time point because it is on an indefinite time scale.

Gregorian day of year

Definition: calendar day that *is on* the Gregorian year of days scale

Note: This is a relative time point because it is on a finite time scale.

Necessity: Each Gregorian day of year *corresponds to* a set *of* Gregorian days.

Note: In general each Gregorian day of year corresponds to one calendar day in each Gregorian year but Gregorian day of year 366 occurs only in leap years.

Gregorian day of month

Definition: calendar day that *is on* the Gregorian month of days scale

Note: This is a relative time point because it is on a finite time scale.

Necessity: Each Gregorian day of month *corresponds to* a set *of* Gregorian days.

Note: In general each Gregorian day of month corresponds to one calendar day in each Gregorian month but Gregorian day of month 29, 30, and 31 do not occur in every Gregorian month.

Because of the cyclic usage of the finite time scales associated with calendars, the names of months, days of the week, and holidays designate many time intervals and so are general concepts. However, these names are traditionally capitalized like proper names and also seem like individual concepts. This illustrates a dilemma: the names of calendar periods are homonyms. The multiple meanings of such names can be understood by considering that each such name can refer to a set of time points collectively, to any member of such a set, or to a unique time point on a finite time scale. For example, “Tuesday” can refer to the set of all Tuesdays (time points), to a member of the set of time points, or to a unique time point on the finite week-of-days scale. To resolve the homonym when using this specification, when a name of a calendar period is used, it is, by default, assumed to designate a general concept of time points. Such usage commonly involves quantifiers, such as “every Tuesday” or “next Tuesday” that select particular time points, and reference particular time intervals, like quantified variables in first order predicate logic. When the unique time point is meant, the name is mentioned (by quoting it) and is qualified as referring to a time point, as in “the time point ‘Tuesday’.” Calendar period names that are treated in this way include the month names, the day-of-week names, and recurring holiday or anniversary names.

Some holidays, like Easter and Ramadan, recur irregularly, so additional information, such as an ephemeris, is required to resolve the name to particular Gregorian calendar dates. Formalizing such definitions is beyond the scope of this specification.

The following defines the common names for Gregorian month of years as individual concepts because they identify specific months on the Gregorian year-of-months scale.

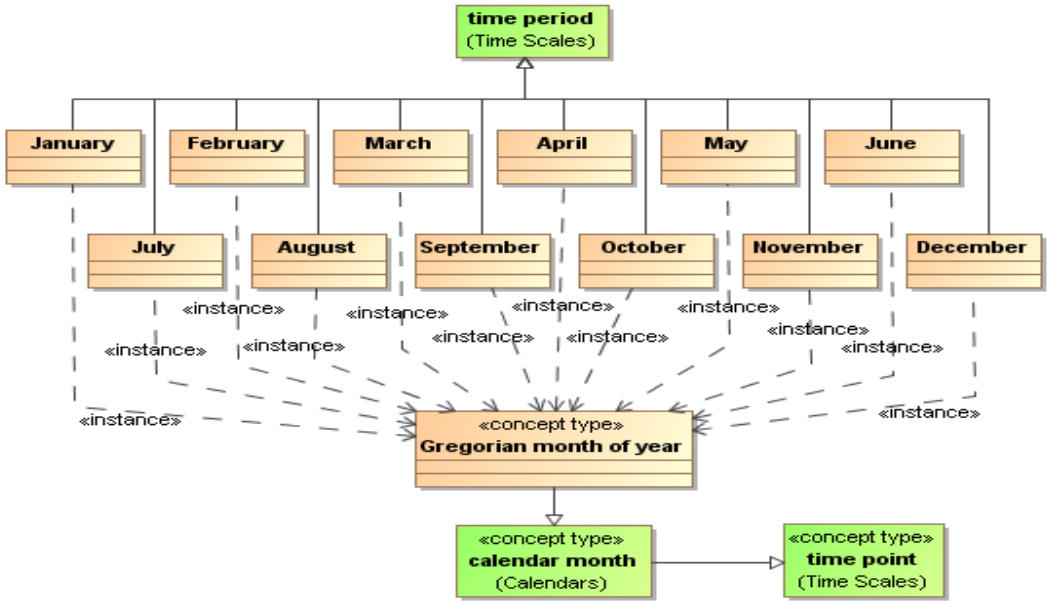

Figure 9.15 - Gregorian Months

January

Source:	ISO 8601 (Table 1)
Definition:	Gregorian month of year *in* sequence position 1 *of* the Gregorian year-of-months scale
Necessity:	The first Gregorian day of month *of* each January *is met by* the last Gregorian day of month *of* a December.
Necessity:	Each January *starts* a Gregorian year.
Necessity:	January *has* 31 Gregorian days of month.
Necessity:	The duration *of* each January *is* 31 days.
Note:	"January 2008" and "2008 month 01" are expressions for a calendar date
Note:	"January, 2008" is an expression for a calendar date for a Gregorian month of year using a reference scheme involving a Gregorian month and a calendar year.

February

Source:	ISO 8601 (Table 1)
Definition:	Gregorian month of year *in* sequence position 2 *of* the Gregorian year-of-months scale
Necessity:	The first Gregorian day of month *of* each February *is met by* the last Gregorian day of month *of* a January.
Necessity:	If the Gregorian year *is* a leap year, then February *has* 29 Gregorian days of month.
Necessity:	If the Gregorian year *is* a common year, then February *has* 28 Gregorian days of month.
Necessity:	The duration *of* each February that *is of* a leap year *is* 29 days.
Necessity:	The duration *of* each February that *is of* a common year *is* 28 days.

Note: The rules for leap years were established by Pope Gregory XIII in [Inter Gravissimas]. These rules were eventually adopted by various civil governments and incorporated into [ISO 8601].

March

Source: ISO 8601 (Table 1)

Definition: Gregorian month of year *in* sequence position 3 *of* the Gregorian year-of-months scale

Necessity: The first Gregorian day of month *of* each March *is met by* the last Gregorian day of month *of* a February.

Necessity: March *has* 31 Gregorian days of month.

Necessity: The duration *of* each March *is* 31 days.

April

Source: ISO 8601 (Table 1)

Definition: Gregorian month of year *in* sequence position 4 *of* the Gregorian year-of-months scale

Necessity: The first Gregorian day of month *of* each April *is met by* the last Gregorian day of month *of* a March.

Necessity: April *has* 30 Gregorian days of month.

Necessity: The duration *of* each April *is* 30 days.

May

Source: ISO 8601 (Table 1)

Definition: Gregorian month of year *in* sequence position 5 *of* the Gregorian year-of-months scale

Necessity: The first Gregorian day of month *of* each May *is met by* the last Gregorian day of month *of* an April.

Necessity: May *has* 31 Gregorian days of month.

Necessity: The duration *of* each May *is* 31 days.

June

Source: ISO 8601 (Table 1)

Definition: Gregorian month of year *in* sequence position 6 *of* the Gregorian year-of-months scale

Necessity: The first Gregorian day of month *of* each June *is met by* the last Gregorian day of month *of* a May.

Necessity: June *has* 30 Gregorian days of month.

Necessity: The duration *of* each June *is* 30 days.

July

Source: ISO 8601 (Table 1)

Definition: Gregorian month of year *in* sequence position 7 *of* the Gregorian year-of-months scale

Necessity: The first Gregorian day of month *of* each July *is met by* the last Gregorian day of month *of* a June.

Necessity: July *has* 31 Gregorian days of month.

Necessity: The duration *of* each July *is* 31 days.

August

Source:	ISO 8601 (Table 1)
Definition:	Gregorian month of year *in* sequence position 8 *of* the Gregorian year-of-months scale
Necessity:	The first Gregorian day of month *of* each August *is met by* the last Gregorian day of month *of* a July.
Necessity:	August *has* 31 Gregorian days of month.
Necessity:	The duration *of* each August *is* 31 days.

September

Source:	ISO 8601 (Table 1)
Definition:	Gregorian month of year *in* sequence position 9 *of* the Gregorian year-of-months scale
Necessity:	The first Gregorian day of month *of* each September *is met by* the last Gregorian day of month *of* an August.
Necessity:	September *has* 30 Gregorian days of month.
Necessity:	The duration *of* each September *is* 30 days.

October

Source:	ISO 8601 (Table 1)
Definition:	Gregorian month of year *in* sequence position 10 *of* the Gregorian year-of-months scale
Necessity:	The first Gregorian day of month *of* each October *is met by* the last Gregorian day of month *of* a September.
Necessity:	October *has* 31 Gregorian days of month.
Necessity:	The duration *of* each October *is* 31 days.

November

Source:	ISO 8601 (Table 1)
Definition:	Gregorian month of year *in* sequence position 11 *of* the Gregorian year-of-months scale
Necessity:	The first Gregorian day of month *of* each November *is met by* the last Gregorian day of month *of* a October.
Necessity:	November *has* 30 Gregorian days of month.
Necessity:	The duration *of* each November *is* 30 days.

December

Source:	ISO 8601 (Table 1)
Definition:	Gregorian month of year *in* sequence position 12 *of* the Gregorian year-of-months scale
Necessity:	The first Gregorian day of month *of* each December *is met by* the last Gregorian day of month *of* a November.
Necessity:	December *has* 31 Gregorian days of month.
Necessity:	The duration *of* each December *is* 31 days.

9.5.6 Week Calendar

The week calendar has been used for centuries, separate from and in combination with the Gregorian calendar, even though they are incommensurate. This calendar supports human discourse using weekday names such as “Monday”, “Tuesday”, and so forth. References to specific weeks are by week of year coordinate.

This specification follows [ISO 8601] in defining "Monday" as the first day of the week. Various cultures and religions define other initial week days. Users of this specification are welcome to redefine the weekday concepts according to their preferences.

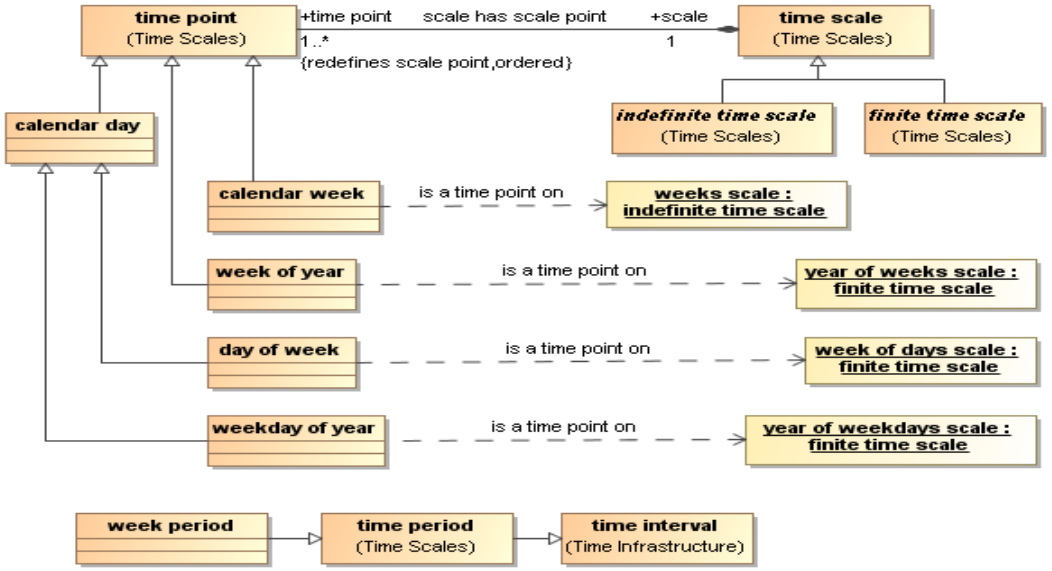

Figure 9.16 - Calendar Week Time Scales, Time Points, and Time Periods

calendar week

Dictionary Basis:	ISO 8601 (2.2.8, 'calendar week')
Definition:	time point that *is on* the week scale and that is defined by a given calendar as 7 consecutive calendar days
Note:	ISO 8601 adds "starting on a Monday" to this definition. This vocabulary drops that phrase because it is culture-specific.
Note:	This is an absolute time point because it is on an indefinite time scale.
Example:	The third calendar week of 2009.

week period

Definition:	time period that starts at a certain time of day at a certain calendar date of the calendar week and ends at the same time of date at the same calendar date of the next calendar week.
Note:	Calendar week is a period that starts and ends as defined by a calendar. Week period starts and ends at any time within a calendar week.
Example:	Tuesday to Tuesday.

day of week

Definition:	calendar day that *is on* the week of days scale
Note:	This is a relative time point because it is on a finite time scale.
Example:	Tuesday is day of week 2

week of year

Definition: calendar week that *is on* the year of weeks scale

Note: This is a relative time point because it is on a finite time scale.

weekday of year

Definition: calendar day that *is on* the year of weekdays scale

The following time scales are defined regarding calendar weeks.

weeks scale

Definition: indefinite time scale *of* granularity 'week' and *of* 'calendar week' time points

Note: No special meaning is assigned to the indices of calendar weeks on the week scale. Therefore, no index origin element or index origin value is specified for this time scale.

week of days scale

Definition: the finite time scale that *subdivides* 'calendar week' into 7 of 'calendar day'

Necessity: The granularity *of* the week of days scale *is* 'day'.

Necessity: The index origin value *of* the week of days scale *equals* 1.

Necessity: The first element *of* the week of days scale *is* the index origin element *of* the week of days scale

year of weeks scale

Definition: the finite time scale that *subdivides* 'calendar year' into 53 of 'week of year'

Note: From [ISO 8601] clause 3.2.2: A calendar year has 52 or 53 weeks of year.

Necessity: The granularity *of* the year of weeks scale *is* 'week'.

Necessity: The index origin value *of* the year of weeks scale *equals* 1.

Necessity: The first element *of* the year of weeks scale *is* the index origin element *of* the year of weeks scale

year of weekdays scale

Definition: the finite time scale that *subdivides* 'calendar year' into 366 of 'calendar day'

Note: The difference between the year of weekdays scale and the year of days scale is the notional alignment of the year of weekdays scale to one of the first seven days of the calendar year, and the fact that the last 3 calendar days of the year of weekdays scale may extend into the next Gregorian year.

Necessity: The granularity *of* the year of weekdays scale *is* 'day'.

Necessity: The index origin value *of* the year of weeks scale *equals* 1.

Necessity: The first element *of* the year of weeks scale *is* the index origin element *of* the year of weeks scale

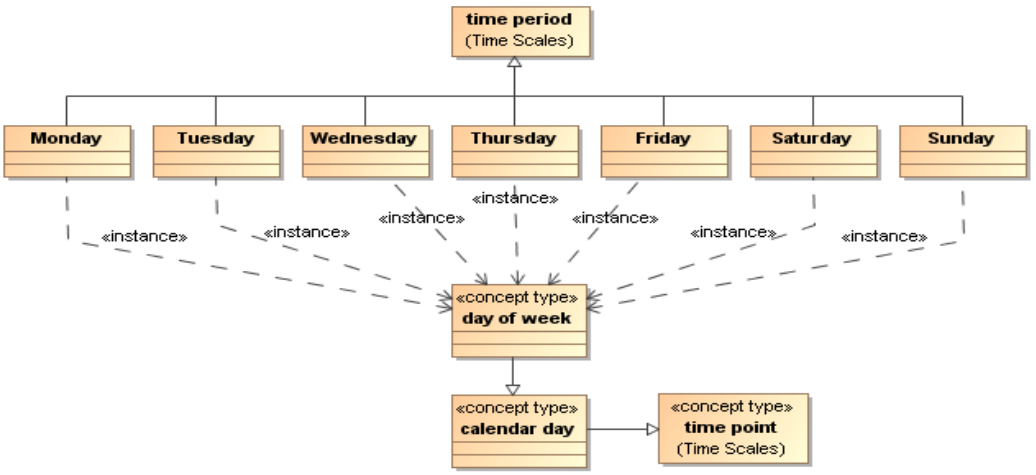

Figure 9.17 - Weekdays

Monday

Source: ISO 8601 (Table 2)

Definition: day of week *in* sequence position 1 *of* the week of days scale

Necessity: Each Monday *is met by* a Sunday.

Tuesday

Source: ISO 8601 (Table 2)

Definition: day of week *in* sequence position 2 *of* the week of days scale

Necessity: Each Tuesday *is met by* a Monday.

Wednesday

Source: ISO 8601 (Table 2)

Definition: day of week *in* sequence position 3 *of* the week of days scale

Necessity: Each Wednesday *is met by* a Tuesday.

Thursday

Source: ISO 8601 (Table 2)

Definition: day of week *in* sequence position 4 *of* the week of days scale

Necessity: Each Thursday *is met by* a Wednesday.

Friday

Source: ISO 8601 (Table 2)

Definition: day of week *in* sequence position 5 *of* the week of days scale

Necessity: Each Friday *is met by* a Thursday.

Saturday

Source: ISO 8601 (Table 2)

Definition: day of week *in* sequence position 6 *of* the week of days scale

Necessity: Each Saturday *is met by* a Friday.

Sunday

Source: ISO 8601 (Table 2)

Definition: day of week *in* sequence position 7 *of* the week of days scale

Necessity: Each Sunday *is met by* a Saturday.

9.5.7 Internet Calendar

Internet Time is the calendar of the Network Time Protocol (NTP), published by the Internet Engineering Task Force (IETF). Virtually all computers and cell phones are synchronized with the NTP, using UDP/IP. Many time dissemination services are available; see ftp://ftp2.bipm.org/pub/tai/scale/TIMESERVICES/timeservices.pdf.

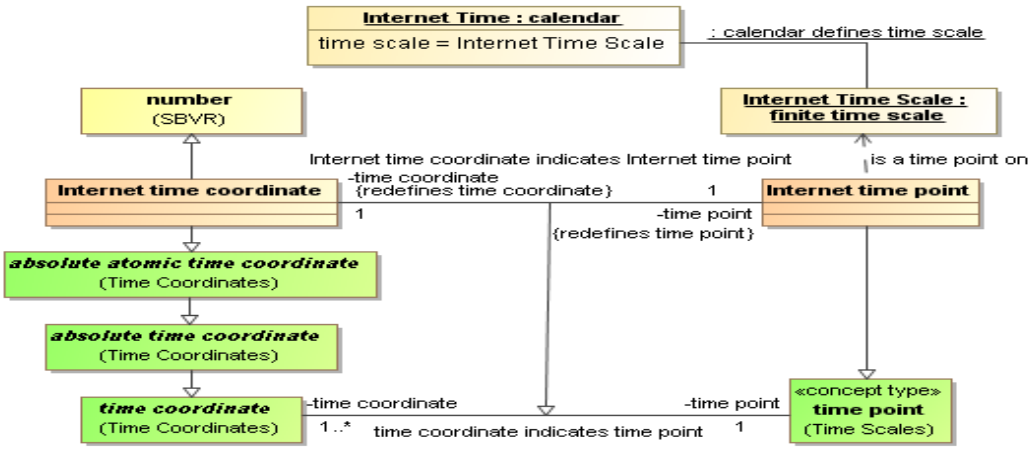

Figure 9.18 - Internet Calendar

Internet Time

Definition: calendar that keeps UTC time and that *uses* the Internet Time Scale

Source: [NTP]

Note: Internet Time is based on UTC but is not necessarily always coincident with it (see [NTP] Appendix E.8 for a fuller explanation of reckoning the Internet Time Scale with UTC). Internet Time accounts for UTC's leap seconds, with a small uncertainty around the time of insertion of a leap second.

Necessity: Accuracy of Internet Time relative to UTC is on the order of 1 millisecond. Stated precision is 200 picoseconds.

Internet Time Scale

Definition: finite time scale whose granularity *is* 2^{-32} seconds and *whose* cardinality *is* 2^{64} and *whose* first 2^{32} time points *correspond to* January 1, 1900 00:00:00 UTC

Note: The data format of NTP is defined in [NTP] section 3.1 and Appendix A. The Internet Time Scale will overflow the 64 bits after about 136 years, in 2036. The IETF is considering a revision of NTP (RFC 5905) that may likely extend its lifetime considerably.

Internet time coordinate

Definition: time coordinate that is a 64-bit unsigned fixed-point number of seconds since January 1, 1900 00:00:00 UTC having a 32 bit integer part and 32 bit fractional part

9.6 Time Tables

These concepts support defining sets of time intervals, called table entries. The table entries can be sequential or overlapping, and at regular or irregular intervals. Time tables with sequential table entries that repeat at regular intervals are called regular time tables. An example would be an airline time table. Time tables that have overlapping table entries or table entries that repeat irregularly are called ad-hoc time tables. For example, conference schedules often have overlapping meetings of varying lengths.

There are two 'time interval' concepts associated with time tables. A 'table entry' is an individual element of the set of time intervals that constitute a time table. A 'time table interval' is the time interval of the time table as a whole. It is the 'sum' or 'convex hull' of the table entries.

The verb concept 'schedule *is for* general situation model' (sub clause 8.3.7) means that an individual situation model that *refines* the general situation model *occurs for* each table entry of a schedule (a role of 'time table'). The verb concept 'occurrence *occurs for* time interval' may be used to say that an occurrence happens for the entire time interval of a time table. For example, a conference meeting might *occur for* a particular table entry of some time table, while the entire conference occurs for the table time period.

See Annex C.4 for a time table example.

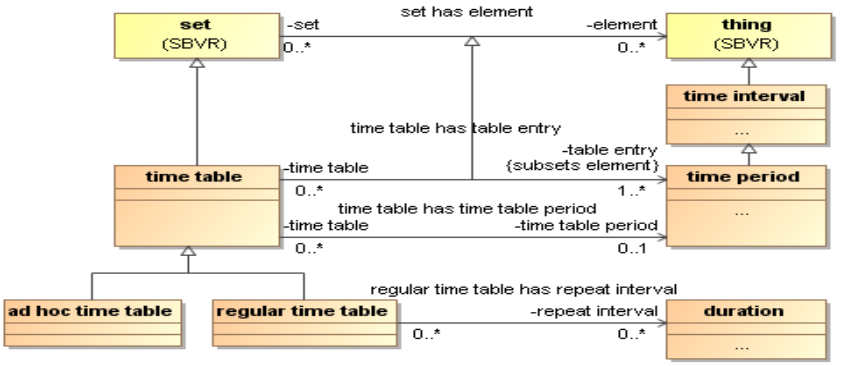

Figure 9.19 - Time Tables

time table

Definition: set *of* table entries

Necessity: Each time table must *have* at least one table entry.

Example: A one hour meeting is scheduled for every Monday at 1pm throughout 2009. The time table time period of the time table is 2009. The first table entry is for Monday, January 5, 2009 from 1pm to 2pm. The repeat interval of the time table is 1 week.

Note: An ad hoc time table may not have any order, hence the general case of "time table" must be a set, not a sequence.

table entry

Concept Type: role

General Concept: time period

Note: The time period may be in the past or the future.

time table period

Concept Type: role

General Concept: time period

Note: This is the time period of the time table as a whole, not of the individual table entries. It is the "convex hull" of all the table entries.

time table *has* time table period

Synonymous Form: time table period *of* time table

Definition: The time entry$_1$ *of* the time table that *precedes* each table entry *of* the time table *plus* the time table$_2$ *of* the time table that *follows* each table entry *of* the time table.

Possibility: A time table period may *be of* more than one time table.

Note: The two Necessities above make time table period the time equivalent of the geometric concept called "convex hull." This is the minimal time period that *includes* all the time table entries of the time table.

Note: In an ad hoc time table, there may be multiple earliest or latest time table entries that *start* or *finish* together.

ad hoc time table

Definition: time table that *has* an extensional definition or *does* not *have* a repeat interval

Description: Each table entry of an ad hoc time table is specified separately.

Note: The term 'extensional definition' is defined in [SBVR] clause 11.1.3.

Example: A meeting time table for meetings on Monday at 1pm to 2pm, Tuesday at 2pm for 30 minutes, and Wednesday at 3pm for an hour – all during the second week of October, 2009.

Note: An ad hoc time table may have time table entries that, although stated explicitly, happen to follow a repeating pattern. For example, an ad hoc meeting might be on Monday, Tuesday, and Wednesday at 1pm for 1 hour.

Example: A (very strange) meeting is scheduled for each day in 2009 whose day of year number is prime.

Note: The example immediately above has an intensional definition, but no repeat interval.

regular time table

Definition: time table that *is* a sequence that *has* an intensional definition with time table entries that repeat according to the repeat interval

Necessity: If a table entry$_1$ of the time table *precedes* a table entry$_2$ of the time table then the index *of* table entry$_1$ *is less than* the index *of* table entry$_2$.

Necessity: If the index *of* some table entry$_2$ *of* the time table is $\underline{1}$ *greater than* the index *of* some table entry$_1$ *of* the time table, then the duration *from* table entry$_1$ *to* table entry$_2$ *is* the repeat interval *of* the time table.

Possibility: Each table entry$_1$ *of* each time table may *meet* some table entry$_2$ *of* the time table.

Description:	A regular time table has time table entries that repeat, unlike an irregular time table of individually-scheduled entries.
Note:	The term ‘intensional definition’ is defined in [SBVR] clause 11.1.3.
Note:	The first Necessity above requires that a regular time table is ordered by its table entries.
Example:	A particular train departs on a daily schedule. The repeat interval is 1 day.
Example:	A meeting scheduled for the third Wednesday of each month period.
Note:	The example immediately above has a repeat interval of “each month period.” At a finer granularity (such as days), the repeat interval varies. But, as given in months, the repeat interval is regular.

regular time table *has* repeat interval

Synonymous Form:	repeat interval *of* regular time table

repeat interval

Concept Type:	role
Definition:	duration between each time table entry *of* a regular time table
Definition:	A repeat interval *is* a nominal duration.

10 Indexical Time Concepts (normative)

"Indexical" is a linguistic concept that refers to terms that make implicit reference to the speaker or the context of the communication. It includes words like "now," "here," "we," etc. This clause defines indexical terms for time periods that are in common business use.

The use of indexical terms in business vocabularies and rules can be ambiguous, and the practice is generally deprecated, but these concepts are needed for some use cases.

10.1 Indexical Characteristics

These unary fact types locate time intervals relative to the fundamental concept 'time interval *is past*'. An alternative design choice would be to specify a fundamental concept 'current time' (or 'now,' or 'now time') as a kind of 'time interval', and then define 'time interval *is past*', 'time interval *is future*', etc., in terms of 'current time'. One of them must be defined; otherwise the definitions are circular. But every time interval has a duration, and defining 'current time' implies specifying its duration. The advantage of making 'time interval *is past*' fundamental is that we need not give a duration for current time.

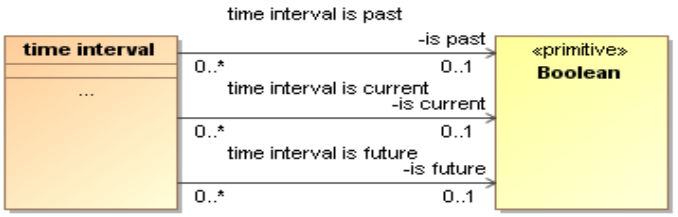

Figure 10.1 - Indexical Characteristics

time interval *is past*

Definition: time interval that *is before* some reference time interval that is defined by context

Note: The reference time interval is sometimes called 'now.' The reference time interval is determined by the context of use of the 'time interval *is past*' verb concept. For a business rule, the reference time interval is the time interval in which the decision to apply and comply with the rule is made. For facts, the reference time interval is one for which a consistent view of the state of the world of interest is to be used in making decisions. That is, 'now' is the time at which the fact or rule is used.

Example: The time interval *identified by* "January 1, 1900" *is past*.

time interval *is current*

Synonymous Form: time interval *is present*

Synonymous Form: time interval *is now*

Definition: time interval that *includes* a time interval$_1$ that *is past* and that *includes* a time interval$_2$ that *is* not *past*

Example: If the contract deadline *is current* …

time interval *is future*

Definition:	time interval that *includes* no time interval that *is in the past*
Necessity:	Each time interval that *is future*, *is after* each time interval that *is past*.
Example:	The supplier may respond to the RFP only if the due date of the RFP *is future*.

These definitions of 'time interval *is past*', 'time interval *is current*', and 'time interval *is future*' are under-specified in the sense that many time intervals (of different durations) fit them. In particular, the verb concept 'time interval *is future*' includes the "now" reference time interval for the verb concept 'time interval *is past*'. Rules that compare time against "now" may be stated more precisely by referencing the indexicals given in sub clause 10.2. For example "if the contract due date *is after* today ..." clearly tests the time interval given by the contract due date against a time interval that has a duration of 1 day and an alignment against the Gregorian calendar, whereas "if the contract due date *is future*" may be interpreted with any "comparison granularity," such as 'second' or 'hour'.

10.2 Indexical Time Intervals

This sub clause defines noun concepts that are indexical references to time intervals.

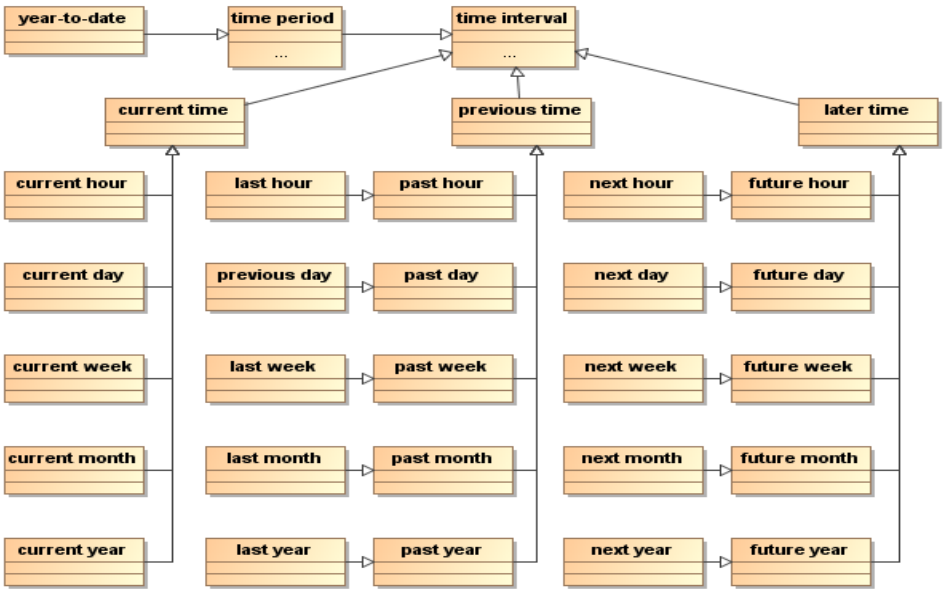

Figure 10.2 - Indexical Time Intervals

current time

Synonym:	present time
Definition:	time interval that *is current*
Note:	Every time interval that overlaps the "reference time interval" for 'time interval *is past*' is a current time (one of many).
Example:	If the "reference time interval" is the current hour, then the calendar day, calendar week, calendar month, calendar year (etc.) that overlap the current hour are all current times.

previous time

Synonym: past time

Definition: time interval that *is past*

later time

Synonym: future time

Definition: time interval that *is future*

current hour

Synonym: this hour

Definition: time interval that *instantiates* some hour of day and *is current*

Example: If “now” is 10:32, then current hour is the time interval denoted as hour of day 10.

last hour

Synonym: previous hour

Definition: time interval that *instantiates* an hour of day and that *meets* current hour

Example: If “now” is 10:32, then last hour is the time interval denoted as hour of day 9.

past hour

Definition: time interval that *instantiates* an hour period and that *meets* some minute of hour that *is current*

Example: If “now” is 10:32, then past hour is the time interval from 9:32 through 10:31.

prior hour

Definition: hour of day that *precedes* the current hour

Definition: hour of day that *is past*

Example: If “now” is 10:32, then one prior hour is the time interval denoted as hour of day 9. Another prior hour is the time interval denoted as hour of day 8.

upcoming hour

Definition: the time interval that *instantiates* an hour of day and that *is met by* the current hour

Example: If “now” is 10:32, then upcoming hour is the time interval denoted as hour of day 11.

next hour

Definition: time interval that *instantiates* an hour period and that *is met by* some minute of hour that *is current*

Example: If “now” is 10:32, then upcoming hour is the time interval from 10:33 through 11:32.

future hour

Definition: hour of day that *is after* the current hour

Definition: hour of day that *is future*

Example: If “now” is 10:32, then one future hour is the time interval denoted as hour of day 11. Another future hour is the time interval denoted as hour of day 12.

current day

Synonym: today

Synonym: this day

Definition: time interval that *instantiates* some calendar day that *is current*

Example: If “now” is July 7 10:32, then current day is the time interval denoted as July 7.

last day

Synonym: yesterday

Synonym: previous day

Definition: time interval that *instantiates* a calendar day and that *meets* current day

Example: If “now” is July 7 10:32, then last day is the time interval denoted as July 6.

past day

Definition: time interval that *instantiates* a day period and that *meets* some minute of hour that *is current*

Example: If “now” is July 7 10:32, then past day is the time interval from July 6 10:32 *through* July 7 10:31.

prior day

Definition: calendar day that *precedes* current day

Example: If “now” is July 7, then one prior day is the time interval denoted by July 6 and another is the time interval denoted by July 5.

upcoming day

Synonym: tomorrow

Definition: time interval that *instantiates* a calendar day and that *is met by* current day

Example: If “now” is July 7, then upcoming day is July 8.

next day

Definition: time interval that *instantiates* a day period and that *is met by* some minute of hour that *is current*

Example: If “now” is July 7 10:32, then next day is the time interval from July 7 10:33 *through* July 8 10:32.

future day

Definition: calendar day that *is after* current day

Definition: calendar day that *is future*

Example: If “now” is July 7, then one future day is the time interval that is denoted by July 8, and another future day is the time interval that is denoted by July 9.

current week

Synonym: this week

Definition: time interval that *instantiates* some calendar week that *is current*

Example: If “now” is week 15 day 3, then current week is the time interval that instantiates week 15.

last week

Synonym: previous week

Definition: time interval that *instantiates* a calendar week and that *meets* current week

Example: If “now” is week 15 day 3, then last week is the time interval that instantiates week 14.

past week

Definition: the time interval that *instantiates* a week period and that *is met by* current day

Example: If “now” is week 15 day 3, then current week is the time interval that is from week 14 day 3 through week 15 day 2.

prior week

Definition: calendar week that *precedes* current week

Definition: calendar week that *is past*

Example: If “now” is week 15 day 3, then one prior week is the time interval that instantiates week 14, and another prior week is the time interval that instantiates week 13.

upcoming week

Definition: the time interval that *instantiates* a calendar week and that *is met by* current week

Example: If “now” is week 15 day 3, then upcoming week is the time interval that instantiates week 16.

next week

Definition: the time interval that *instantiates* a week period and that *is met by* current day

Example: If “now” is week 15 day 3, then next week is the time interval that is from week 15 day 4 *through* week 16 day 3.

future week

Definition: calendar week that *is after* the current week

Definition: calendar week that *is future*

Example: If “now” is week 15 day 3, then one future week is the time interval that instantiates week 16 and another future week is the time interval that instantiates week 17.

current month

Synonym: this month

Definition: the time interval that *instantiates* some calendar month that *is current*

Example: If “now” is July 7, then current month is the time interval that instantiates July.

last month

Synonym: previous month

Definition: the time interval that *instantiates* a calendar month and that *meets* the current month

Example: If “now” is July 7, then last month is the time interval that instantiates June.

past month

Definition: time interval that *instantiates* a month period and that *meets* some day of year that *is current*

Necessity: The duration *of* past month *is* the duration *of* last month.

Note: The previous Necessity addresses the varying duration of calendar months.

Example: If “now” is July 7, then past month is the time interval from June 7 *through* July 6.

Example: If “now” is June 7, then past month is the time interval from May 7 *through* June 6.

prior month

Definition: calendar month that *precedes* the current month

Definition: calendar month that *is past*

Example: If "now" is July 7, then one prior month is the time interval that instantiates June, and another prior month is the time interval that instantiates May.

upcoming month

Definition: the time interval that *instantiates* a calendar month and that *is met by* the current month

Example: If "now" is July 7, then upcoming month is the time interval that instantiates August.

next month

Definition: time interval that *instantiates* a month period and that *is met by* some day of year that *is current*

Necessity: The duration *of* next month *is* the duration *of* current month.

Note: The previous Necessity addresses the varying duration of calendar months.

Example: If "now" is July 7, then next month is the time interval from July 8 through August 7.

Example: If "now" is June 7, then next month is the time interval from June 8 through July 7.

future month

Definition: calendar month that *is after* the current month

Definition: calendar month that *is future*

Example: If "now" is July 7, then one future month is the time interval that instantiates August, and another future month is the time interval that instantiates September.

current year

Definition: this year

Definition: the time interval that *instantiates* some calendar year that *is current*

Example: If "now" is July 11, 2011, then current year is the time interval that instantiates 2011.

last year

Synonym: previous year

Definition: the time interval that *instantiates* a calendar year and that *meets* the current year

Example: If "now" is July 11, 2011, then last year is the time interval that instantiates 2010.

past year

Definition: time interval that *instantiates* a year period and that *meets* some day of year that *is current*

Necessity: The duration *of* past year *is* the duration *of* last year.

Note: The previous Necessity addresses the varying duration of calendar years.

Example: If "now" is July 11 2011, then past year is the time interval from July 11 2010 *through* July 10 2011.

prior year

Definition: calendar year that *precedes* the current year

Definition: calendar year that *is past*

Example: If “now” is July 11 2011, then one prior year is the time interval that instantiates 2010 and another prior year is the time interval that instantiates 2009.

upcoming year

Definition: the time interval that *instantiates* a calendar year and that *is met by* the current year

Example: If “now” is July 11 2011, then upcoming year is the time interval that instantiates 2010.

next year

Definition: time interval that *instantiates* a year period and that *is met by* some day of year that *is current*

Necessity: The duration *of* next year *is* the duration *of* current year.

Note: The previous Necessity addresses the varying duration of calendar years.

Example: If “now” is July 7 2011, then next year is the time interval from July 8 2011 through August 7 2012.

future year

Definition: calendar year that *is after* the current year

Definition: calendar year that *is future*

Example: If “now” is July 7 2011, then one future year is the time interval denoted by 2012 and another future year is the time interval denoted by 2013.

year-to-date

Definition: the time period that *starts on* calendar day 1 *of* the current year and that *ends on* the current day

10.3 Language Tense and Aspect

As discussed in sub clause 7.11, human languages use past, present, and future tenses and incorporate simple, progressive, and perfect aspects. This sub clause provides concepts that enable all these tenses and aspects, in any combination. They extend the relationships between situation models, occurrences, and time that are defined in clause 8.3.

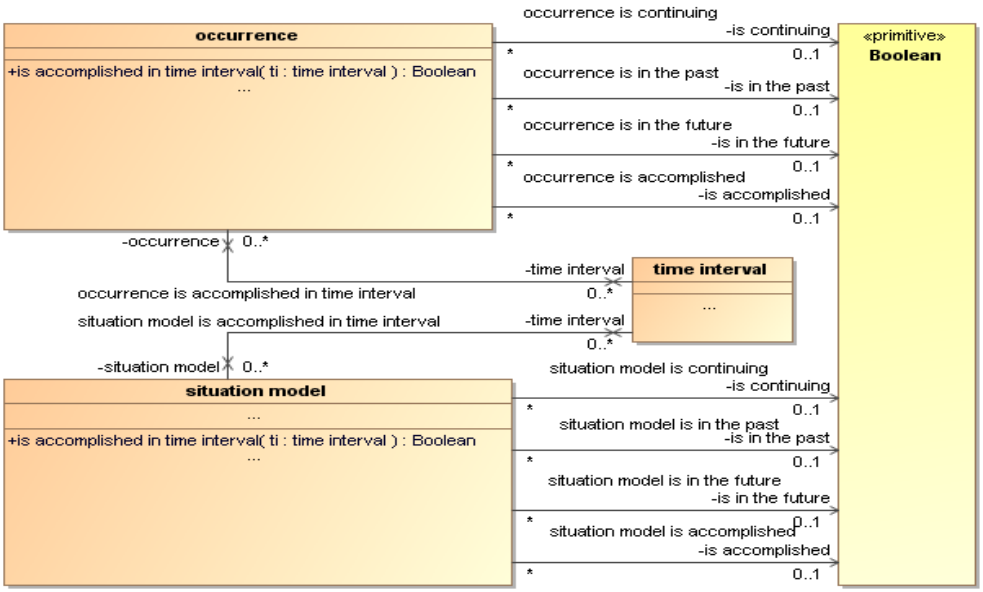

Figure 10.3 - Language Tense and Aspect

The following verb concepts formalize the progressive and perfect language aspects. The concepts are provided for both 'situation model' and 'occurrence'; the former are normally used in guidance statements, while the latter are most useful in facts.

situation model *is continuing*

Definition:	the situation model is unfinished at some reference time interval
Note:	The reference time interval is when a fact is evaluated or a rule is being applied.
Note:	'*situation model is continuing*' indicates the progressive aspect of natural language. It is sometimes called the "continuous aspect."
Example:	If company x is going bankrupt....
Note:	'*Situation model is continuing*' is not the negation of '*situation model is accomplished*' because a situation model may end without being accomplished. Consider that the situation model 'John writes book' in the partial rule "if John writes a book ..." may end without John ever completing the book.
Note:	A situation model may be *is continuing* or *is accomplished* or both or neither, and may also be in the past, present, or future tense. (See table below).

situation model *is accomplished*

Definition:	the situation model has reached a point of completion or perfection at with respect to the "reference time interval" associated with the concept '*time interval is past*'
Note:	The reference time interval is when a fact is evaluated or a rule is being applied.
Example:	If company x has gone bankrupt....
Note:	'*Situation model is accomplished*' is not the negation of '*situation model is continuing*' because a situation model may end without being accomplished. Consider that the situation

model ‘John writes book’ in the partial rule “if John writes a book ...” may end without John ever completing the book.

Note: A situation model may be *is continuing* or *is accomplished* or both or neither, and may also be in the past, present, or future tense. (See table below).

situation model *is accomplished in* time interval

Definition: the situation model reaches a point of completion or perfection at some time interval$_2$ that *is part of* the time interval

Example: If the contract is completed within this year ....

occurrence *is continuing*

Definition: the occurrence is unfinished at some reference time interval

Note: The reference time interval is when a fact is evaluated or a rule is being applied.

Note: ‘occurrence *is continuing*’ indicates the progressive aspect of natural language. It is sometimes called the “continuous aspect.”

Example: Company x is going bankrupt.

Note: ‘Occurrence *is continuing*’ is not the negation of ‘occurrence *is accomplished*’ because an occurrence may end without being accomplished. Consider that the occurrence ‘John writes book’ may end without John ever completing the book.

Note: An occurrence may be *is continuing* or *is accomplished* or both or neither, and may also be in the past, present, or future tense. (See table below).

occurrence *is accomplished*

Definition: the occurrence has reached a point of completion or perfection at with respect to the “reference time interval” associated with the concept ‘time interval *is past*’

Note: The reference time interval is when a fact is evaluated or a rule is being applied.

Example: Company x has gone bankrupt.

Note: ‘Occurence *is accomplished*’ is not the negation of ‘occurrences *is continuing*’ because a state of affairs may end without being accomplished. Consider that the state of affairs ‘John writes book’ may end without John ever completing the book.

Note: An occurrence may be *is continuing* or *is accomplished* or both or neither, and may also be in the past, present, or future tense. (See table below).

occurrence *is accomplished in* time interval

Definition: the occurrence reaches a point of completion or perfection at some time interval$_2$ that *is part of* the time interval

Example: The occurrence “Columbus reaches the new world” *is accomplished in* the 15th Century.

These verb concepts enable formulation of past, present, and future tense propositions. As above, the ‘situation model’ versions of these concepts are most useful in guidance statements, while the ‘occurrence’ versions are intended for use in facts.

situation model *is in the past*

Definition: the situation model *occurs throughout* some time interval that *is in the past*

Example: If the customer has previously failed to pay his bill ....

Note: Whether a situation model *is in the past* may be inferred when a situation model is located in time via any of the verb concepts given in clause 8.3.5, such as "situation model$_1$ *is before* situation model$_2$."

situation model *occurs now*

Definition: the situation model *occurs for* some time interval that *is current*

Example: "If the bill is due now" (which might be formulated as "if the bill is due occurs now").

situation model *is in the future*

Definition: the situation model *occurs throughout* some time interval that *is in the future*

Example: "If President Obama will write his memoirs," which might be formulated as "If President Obama writes his memoirs *in the future*."

Note: Whether a situation model *is in the future* may be inferred when a situation model is located in time via any of the verb concepts given above, such as "situation model$_1$ *is before* situation model$_2$."

occurrence *is in the past*

Definition: the occurrence *occurs throughout* some time interval that *is in the past*

Example: The reign of Alexander the Great *is in the past*.

Note: Whether an occurrence *is in the past* may be inferred when an occurrence is located in time via any of the verb concepts given in clause 8.3.3, such as "occurrence$_1$ *is before* occurrence$_2$".

occurrence *occurs now*

Definition: the occurrence *occurs for* some time interval that *is current*

Example: That EU-Rent is in business *occurs now* (which means the same as "EU-Rent is in business now").

occurrence *is in the future*

Definition: the occurrence *occurs throughout* some time interval that *is in the future*

Example: "President Obama writes his memoirs" *is in the future*.

Note: Whether a state of affairs *is in the future* may be inferred when an occurrence is located in time via any of the verb concepts given in sub clause 8.3.3, such as "occurrence$_1$ *is before* occurrence$_2$."

This specification defines vocabulary fact types in the present tense. The following table gives examples of how other tenses and aspects can be formulated. To show the range of expression supported by this vocabulary, some examples reference specific time intervals, while others leave unstated the time interval that an occurrence *is continuing* or *is accomplished*.

This table assumes a domain vocabulary verb concept "John *writes* book". The examples are given as facts, and hence are formulated using the 'occurrence' version of the verb concepts listed above.

Table 10.1 - Formulation of Verb Tenses and Aspects

Simple Aspect		
Tense	Example	Formulation
past	John *wrote* a book	the occurrence (that John *writes* a book) *is in the past*
present	John *writes* a book	John *writes* a book
future	John *will write* a book	the occurrence (that John *writes* a book) *is in the future*
Progressive Aspect		
Tense	Example	Formulation
past	John *was writing* a book last year	the occurrence (that John *writes* a book) *holds within* last year and *is continuous*
present	John *is writing* a book	the occurrence (that John *writes* a book) *is continuous*
future	John *will be writing* a book *during* January 2011 *through* June 2012	the occurrence (that John *writes* a book) *is in the future* and *holds within* the time period (January 2011 *through* June 2012)
Perfect Aspect		
Tense	Example	Formulation
past	John *had written* a book *before* 2009	the occurrence (that John *writes* a book) *is in the past* and *is accomplished at* some time point that *is before* 2009
present	John *has written* a book	the occurrence (that John *writes* a book) *is accomplished*
future	John *will have written* a book	the occurrence (that John *writes* a book) *is in the future* and *is accomplished*
Progressive and Perfect		
Tense	Example	Formulation
past	John *had been writing* a book	the occurrence (that John *writes* a book) *is in the past* and *is continuous* and *is accomplished*
present	John *has been writing* a book	the occurrence (that John *writes* a book) *is continuous* and *is accomplished*
future	John *will have been writing* a book *by* 2012	the occurrence (that John *writes* a book) *is in the future* and *is continuous* and *is accomplished at* some time point that *is before* 2012

At the time of writing this document, the example "John *will be writing* a book *during* January 2021 *through* June 2022" is in the future. Nevertheless, the formulation includes the apparently redundant "*is in the future*" to express the future tense of the statement even after 2022.

The formulation of "John *was writing* a book last year" excludes "*is in the past*" because "last year" applies at all times.

11 Duration Values (normative)

Duration values are amounts of time stated in terms of one or more time units. For example, "60 seconds" or "1 minute". The concept 'duration value', and related concepts, specialize 'quantity value'(Annex D.2.3) and its related concepts. These concepts are restated here for clarification and to bring them into this normative text.

In this specification, a precise duration value *quantifies* a duration. The key difference between 'duration value' and 'duration' is that a single duration may be quantified by multiple precise duration values. For example, "60 seconds" and "1 minute" quantify the same duration: the two duration values are *equivalent*.

Complexity arises with duration values that use the nominal time units 'month' and 'year' because the number of calendar days varies among calendar months, and because some calendar years incorporate leap days. For example, "1 year" is equivalent to "12 months" but it is unclear in everyday usage how "12 months" compares to "365 days". To help answer the question, this clause introduces the concept of 'duration value set'. A duration value set specifies a set of duration values that are jointly considered equivalent to a nominal duration value. For example, "1 month" is any of {28 days, 29 days, 30 days, 31 days}.

Furthermore, this clause specifies common arithmetic and comparison operations on nominal duration values defined as duration value sets. This helps to define what expressions such as "3 months" or "3 months *plus* 3 days" mean. The advantage of this approach is that it clarifies the results of comparisons such as "3 months < 90 days."

11.1 Duration Values

duration value

Definition:	precise duration value or nominal duration value
Necessity:	Each duration value *has* at least one atomic duration value.
Note:	A duration value can be either atomic or compound (see clause 11.2.1) and either nominal or precise (clauses 11.2.2 and 11.2.3).
Example:	45 seconds, 1 year 3 days

11.1.1 Atomic and Compound Duration Values

Duration values can be either atomic (have just one component, such as 10 minutes) or be compound (a combination of multiple atomic duration values, such as 1 year 5 months). Atomic duration values consist of a number and a time unit, such as "4 weeks." Compound duration values comprise multiple atomic duration values. For example, "3 years 5 months".

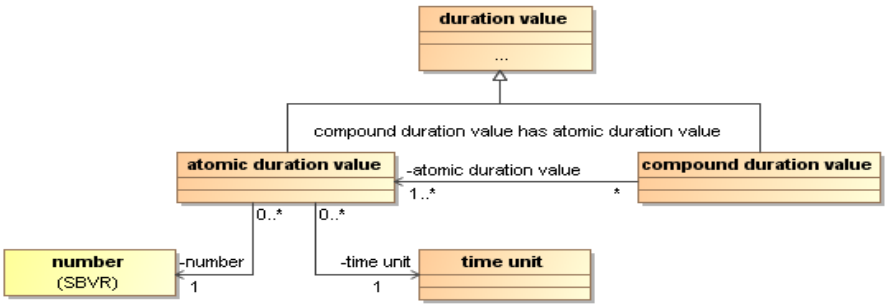

Figure 11.1 - Duration Values

atomic duration value

Definition: number and time unit together giving magnitude of a duration

Dictionary Basis: VIM 1.19 'quantity value'

Example: 55 seconds is an atomic duration value

atomic duration value *has* number

Definition: if the atomic duration value *is* a precise atomic duration value, then the number is the ratio of the duration *quantified by* the atomic duration value to the time unit of the atomic duration value

Definition: if the atomic duration value *is* a nominal atomic duration value, then the number is the ratio of exactly one of the elements of the duration value set that *is specified by* the atomic duration value to the time unit of the atomic duration value

Note: In the general case, the number is a mathematical real or complex number. Because the number is a ratio, rational fractions are commonly used in stating duration values. Thus, it is meaningful to say a task took 2.5 days to complete. Fractional numbers are not defined for nominal atomic duration values (except for ½ year, ¼ year, and ¾ year), because they have no clear meaning.

Example: 2.5 years, 5.6318 seconds

Note: When the number is a non-negative integer, it may be thought of as a count of the time units in the duration value. But that view only applies to certain measurement techniques, such as the count of ticks of a clock.

Example: 8 months

Possibility: The number *is less than* 0.

Note: Although there are no negative durations, the number of an atomic duration value may be negative. A duration value may *quantify* a (positive) duration even though a component atomic duration value is negative. Typically, a negative atomic duration value arises as an intermediate result of a subtraction. "1 hour 12 minutes - 14 minutes *equals* 1 hour –2 minutes", which *quantifies* the same duration that is *quantified by* "58 minutes".

atomic duration value *has* time unit

Definition: if the atomic duration value *is* a precise atomic duration value, then the time unit is the reference duration to which the ratio of the duration *quantified by* the atomic duration value is taken

Definition: if the atomic duration value *is* a nominal atomic duration value, then the time unit is the reference duration to which the ratio of exactly one element of the duration value set *specified by* the atomic duration value is taken

Example: "45 minutes" has the time unit 'minute'

compound duration value

Definition: duration value that is a sum of two or more atomic duration values that *have* different time units

Example: "2 hours 20 minutes" *quantifies* the duration that may also be *quantified* as "140 minutes"

duration value *has* atomic duration value

Definition: the atomic duration value is one of the summands of the duration value

Example: 1 hour 5 minutes 3 seconds is a compound duration value that is composed of three atomic duration values: 1 hour, 5 minutes, 3 seconds

11.1.2 Precise Duration Values

Time units are either precise (such as seconds) or nominal (that is years, which can be either 365 days or 366 days; and months, which can be 28 days, 29 days, 30 days, or 31 days). Duration values are also nominal or precise according to whether they use nominal or precise time units.

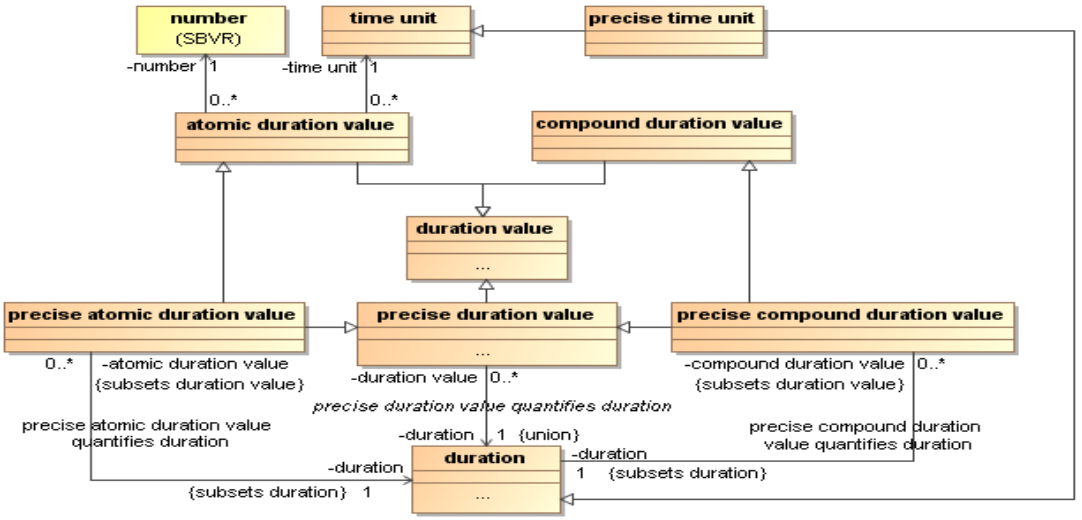

Figure 11.2 - Precise Duration Values

precise duration value

Definition: precise atomic duration value or precise compound duration value

Example: 5 hours

Example: 3 days 5 hours

precise atomic duration value

Definition: number and time unit together giving magnitude of a duration

Dictionary Basis:	VIM 1.19 ‘quantity value’
Definition:	atomic quantity value that *has* a precise time unit
Note:	The duration *quantified by* a precise atomic duration value is the duration whose ratio to the time unit is the number.
Example:	30 seconds

precise compound duration value

Definition:	duration value that is the sum of two or more precise atomic duration values that *have* different time units
Example:	5 minutes 30 seconds

Each precise time unit (i.e., the time units ‘second,’ ‘minute,’ ‘hour,’ ‘day,’ and ‘week’) is defined as *quantifying* a multiple of ‘second’ using the pattern ‘the precise time unit that *quantifies* <some number of> seconds’. Thus, every precise atomic duration value (i.e., an atomic duration value that uses one of those time units) *quantifies* a duration that is some multiple of ‘seconds’. For example, ‘3 hours’ *quantifies* a duration of 10 800 seconds.

precise atomic duration value *quantifies* duration

Synonymous Form:	duration *is quantified by* precise atomic duration value
Definition:	the ratio of the duration to the time unit *of* the precise atomic duration value *is* the number *of* the precise atomic duration value
Example:	“2 seconds” *quantifies* a duration that is twice the duration of the time unit ‘second’
Example:	“1 minute 3 seconds” *quantifies* a duration that is 63 times the duration of the time unit ‘second’

Precise compound duration values *quantify* durations via a computation that can be summarized as “*quantify* all the atomic duration values of the precise compound duration value as durations, and then sum them”. For example, 2 hours 30 minutes 20 seconds *quantifies* a duration of ‘9 020 seconds’.

precise compound duration value *quantifies* duration

Synonymous Form:	duration *is quantified by* precise compound duration value
Definition:	the duration *is* the sum of the durations that *are quantified by* each precise atomic duration value *of* the precise compound duration value
Example:	12 weeks 3 days *quantifies* the duration ‘8 380 800 seconds’

11.1.3 Nominal Duration Values

Nominal duration values are distinguished from precise nominal duration values because a nominal duration value is one of several durations as defined by a calendar. For example, the compound nominal duration value “1 year 1 day” is any of {366 days, 367 days} because 1 year plus 1 day could be either of those.

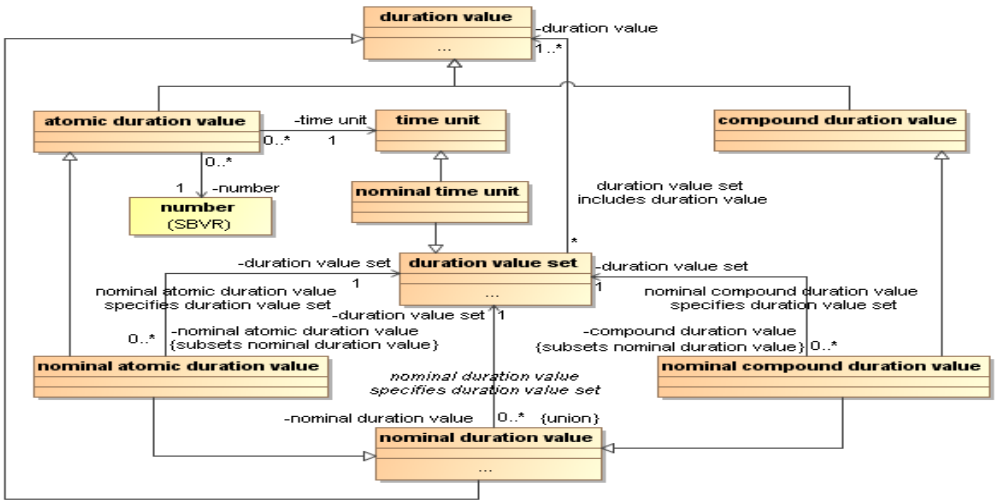

Figure 11.3 - Nominal Duration Values

nominal duration value

Definition: nominal atomic duration value or nominal compound duration value

Necessity: The nominal duration value *is* the range *of* a time interval *identified by* a time period *of* a time calendar.

Example: 5 months, for example from February through June

Example: 2 years 6 months, for example from January 1990

nominal atomic duration value

Definition: number and nominal time unit together that *specify* a duration value set

Note: See clauses 11.5 and 11.6 for the detailed definition of this concept.

Example: 30 months

nominal compound duration value

Definition: compound quantity value in which at least one atomic quantity value *is* a nominal atomic duration value

Possibility: An atomic duration value *of* the nominal compound duration value *is* a precise atomic duration value.

Example: 1 year 1 day

Each nominal time unit (i.e., the time units 'year' and 'month') is defined as *specifying* two or more choices among different numbers of 'days' using the pattern 'the nominal time unit that *specifies* {<number1> days, <number2> days, …, <numbern> days}'. This captures the idea that a year is either 365 days or 366 days, and a month is anywhere from 28 to 31 days.

nominal atomic duration value *specifies* duration value set

Synonymous Form:	duration value set *is specified by* nominal atomic duration value
Definition:	the duration value set is a function of the nominal time unit *of* the nominal atomic duration value and the number *of* the nominal atomic duration value, and that function depends upon the nominal time unit
Note:	The meaning of this verb concept is further defined in specializations, two which are defined in clauses 11.5 and 11.6: 'year value *specifies* duration value set' and 'month value *specifies* duration value set'. Other vocabularies can add their own for other nominal time units.
Example:	2 years *specifies* {730 days, 731 days} because the nominal time unit 'year' *specifies* the duration value set {365 days, 366 days} and there are no two consecutive leap years

Unlike precise atomic duration values, a nominal atomic duration value is not a simple multiple of the duration values of the duration value set *specified by* the nominal time unit of the nominal atomic duration value. For example, 2 years does not *quantify* "2 * 366 days" because, in the Gregorian calendar, two successive years cannot both be leap years. Thus, 2 years *specifies* one of {365 + 365 days, 365 + 366 days}. Clauses 11.5 and 11.6 formally define this for the 'year' and 'month' nominal time units.

A nominal compound duration value comprises two or more nominal atomic duration values. Each of these nominal atomic duration values *specifies* a duration value set, as described above. The entire nominal compound duration value *specifies* a duration value set that is the summation of the individual duration value sets. The summation is computed by pairwise addition of each of the duration values sets that *are quantified by* the nominal atomic duration values. Adding two duration value sets is defined by the verb concept 'duration set_3 = duration set_1 + duration set_2' in sub clause 11.4.

nominal compound duration value *specifies* duration value set

Synonymous Form:	duration value set *is specified by* nominal compound duration value
Definition:	the duration value set *is* the sum of the duration value sets that *are specified by* each atomic duration value *of* the nominal compound duration value
Example:	14 months 3 days *specifies* the duration value set {427 days, 428 days, 429 days, 430 days, 431 days}

11.2 Duration Value Arithmetic

Addition and subtraction of duration values, and multiplication and division of duration values by scalar numbers, is defined in terms of the corresponding operations on the individual components of the duration values. For example, "1 year 5 months + 8 months 8 days" produces "1 year 13 months 8 days". This avoids the complexities of mixed-base arithmetic, which are not resolvable in the case of nominal duration values. (As an example of those complexities, consider that "14 days + 14 days" might be equivalent to either "28 days" or "1 month" depending upon the particular month.)

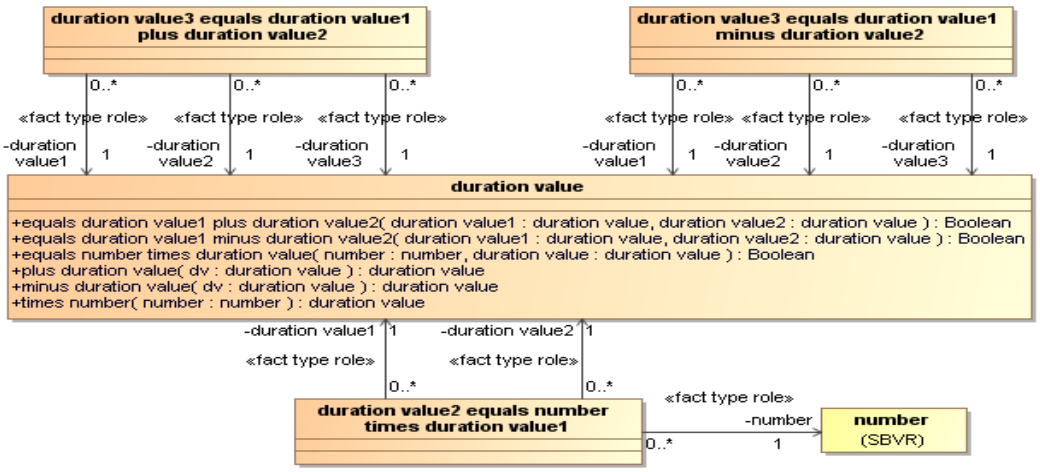

Figure 11.4 - Duration Value Arithmetic

duration value$_3$ *equals* duration value$_1$ *plus* duration value$_2$

Synonymous Form: duration value$_1$ *plus* duration value$_2$

Synonymous Form: duration value$_3$ = duration value$_1$ + duration value$_2$

Synonymous Form: duration value$_1$ + duration value$_2$

Definition: each atomic duration value$_3$ *of* duration value$_3$ equals the sum of the number$_1$ *of* an atomic duration value$_1$ *of* duration value$_1$ and either the number$_2$ *of* some atomic duration value$_2$ *of* duration value$_2$ that *has* the same time unit, or 0 if there *does* not *exist* an atomic duration value$_2$ *of* duration value$_2$ that *has* the same time unit

Note: This does not use “carries” among atomic duration values of different time units, because they don’t work for nominal time units. The numbers of the atomic duration values that comprise duration value$_3$ may be greater than defined in the corresponding time unit.

Example: 6 years 367 days 4 hours 61 minutes *equals* 5 years 3 days 4 hours 3 minutes *plus* 1 year 364 days 58 minutes

Note: Tools may represent the results of duration value addition using mixed-base “carries” when practical.

Example: 1 hour 80 minutes *equals* 1 hour 35 minutes *plus* 45 minutes. A tool may choose to display this result as 2 hours 20 minutes.

duration value$_3$ *equals* duration value$_1$ *minus* duration value$_2$

Synonymous Form: duration value$_1$ *minus* duration value$_2$

Synonymous Form: duration value$_3$ = duration value$_1$ - duration value$_2$

Synonymous Form: duration value$_1$ - duration value$_2$

Definition: each atomic duration value$_3$ *of* duration value$_3$ equals the number$_1$ *of* an atomic duration value$_1$ *of* duration value$_1$ minus either the number$_2$ *of* some atomic duration value$_2$ *of* duration value$_2$ that *has* the same time unit, or 0 if there *does* not *exist* an atomic duration value$_2$ *of* duration value$_2$ that *has* the same time unit

Possibility: The number *of* some atomic duration value *of* duration value$_3$ may be negative.

Note: This does not use "borrows" among atomic duration values of different time units, because they don't work for nominal time units. Negative atomic duration values may occur.

Example: 1 year -5 days *equals* 1 year 45 days *minus* 50 days

duration value$_2$ *equals* number *times* duration value$_1$

Synonymous Form: duration value *equals* duration value *times* number

Synonymous Form: number *times* duration value

Synonymous Form: duration value *times* number

Synonymous Form: duration value = number * duration value

Synonymous Form: duration value = duration value * number

Synonymous Form: number * duration value

Synonymous Form: duration value * number

Definition: each atomic duration value$_1$ *of* duration value$_1$, multiplied by the given number equals some atomic duration value$_2$ *of* duration value$_2$

Example: 5 days *quantifies* the duration that *equals* 5 *times* 1 day

Possibility: The number is negative.

Example: -5 days

Note: Negative duration values arise from arithmetic formulae. However, a negative duration value does not *quantify* any duration.

Possibility: If duration value$_1$ *is* a precise duration value then the number is fractional.

Example: 5.5 days *quantifies* the duration that *equals* 5.5 *times* 1 day

Necessity: If duration value$_1$ *is* a nominal duration value and the number is fractional then the time unit *of* each atomic duration value *of* the duration value *is* not 'month' and the fractional part of the number is ¼, ½, 2/4, or ¾.

Necessity: 3 months *equals* ¼ *times* 'year.'

Necessity: 6 months *equals* ½ *times* 'year.'

Necessity: 6 months *equals* 2/4 *times* 'year.'

Necessity: 9 months *equals* ¾ *times* 'year.'

Note: This specification defines only the fractional nominal duration values ¼ year, ½ year, 2/4 year, and ¾ year because these are in common business use and they equal an integral number of months. This specification does not support any fractional 'month' duration values because such fractions are not in common use and because they are not equal to an integral number of years or days.

Example: 5.5 years *quantifies* the duration that *equals* 5.5 *times* 1 year

11.3 Duration Value Comparison

Comparison of duration values is defined in terms of the same operations on the *quantified* durations or *specified* duration value sets. The benefit of the unusual semantic for nominal duration values is that these comparisons have useful results for many nominal duration values. For example, the expression "1 year 1 day > 365 days" is true for both possible duration values that are *specified by* 1 year 1 day.

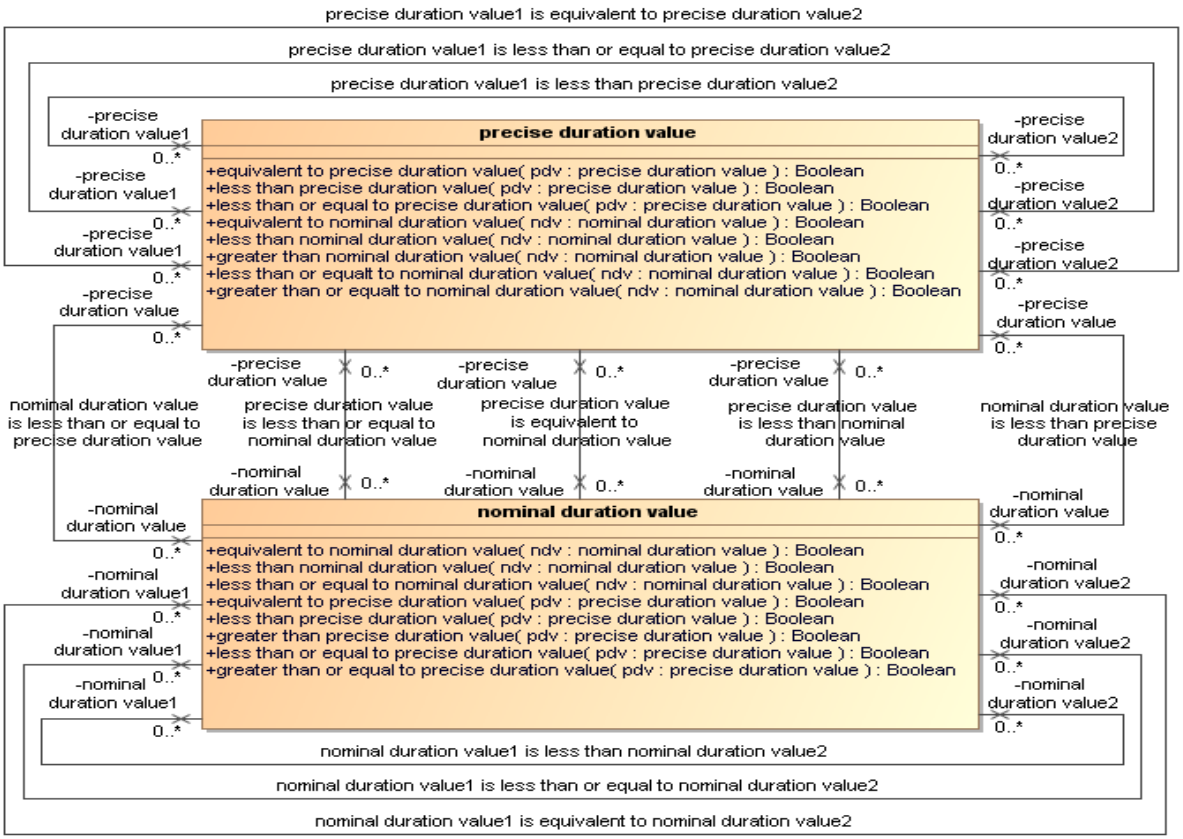

Figure 11.5 - Duration Value Comparison

precise duration value$_1$ *is equivalent to* precise duration value$_2$

Synonymous Form: precise duration value$_1$ *equals* precise duration value$_2$

Synonymous Form: precise duration value$_1$ = precise duration value$_2$

Definition: precise duration value$_1$ *quantifies* duration$_1$ and precise duration value$_2$ *quantifies* duration$_2$ and duration$_1$ = duration$_2$

Example: “3 days 12 hours” *is equivalent to* “84 hours”

nominal duration value$_1$ *is equivalent to* nominal duration value$_2$

Synonym: nominal duration value$_1$ *equals* nominal duration value$_2$

Synonym: nominal duration value$_1$ = nominal duration value$_2$

Definition: nominal duration value$_1$ = duration value set$_1$ and nominal duration value$_2$ = duration value set$_2$ and duration value set$_1$ = duration value set$_2$

Example: “1 month” *is equivalent to* “1 month”

Example: “1 year 1 day” *is* not *equivalent to* “366 days”

precise duration value ***is equivalent to*** **nominal duration value**

Synonym:	precise duration value *equals* nominal duration value
Synonym:	precise duration value = nominal duration value
Synonym:	nominal duration value *is equivalent to* precise duration value
Synonym:	nominal duration value *equals* precise duration value
Synonym:	nominal duration value = precise duration value
Definition:	nominal duration value *quantifies* a duration value set and precise duration value *quantifies* a duration that = some duration *of* the duration value set
Example:	"28 days" *is equivalent to* "1 month"

precise duration value$_1$ ***is less than or equal to*** **precise duration value$_2$**

Synonymous Form:	precise duration value$_2$ *is greater than or equal to* precise duration value$_1$
Synonymous Form:	precise duration value$_1$ ≤ precise duration value$_2$
Synonymous Form:	precise duration value$_2$ ≥ precise duration value$_1$
Definition:	precise duration value$_1$ *quantifies* duration$_1$ and precise duration value$_2$ *quantifies* duration$_2$ and duration$_1$ ≤ duration$_2$
Example:	"1 hour 30 minutes" *is less than or equal to* "2 days 30 minutes"

nominal duration value$_1$ ***is less than or equal to*** **nominal duration value$_2$**

Synonymous Form:	nominal duration value$_2$ *is greater than or equal to* nominal duration value$_1$
Synonymous Form:	nominal duration value$_1$ ≤ nominal duration value$_2$
Synonymous Form:	nominal duration value$_2$ ≥ nominal duration value$_1$
Definition:	nominal duration value$_1$ *quantifies* duration value set$_1$ and nominal duration value$_2$ *quantifies* duration value set$_2$ and duration value set$_1$ ≤ duration value set$_2$
Example:	"1 month 1 day" *is less than or equal to* "1 month 2 days"

precise duration value ***is less than or equal to*** **nominal duration value**

Synonymous Form:	precise duration value ≤ nominal duration value
Synonymous Form:	nominal duration value *is greater than or equal to* precise duration value
Synonymous Form:	nominal duration value ≥ precise duration value
Definition:	precise duration value *quantifies* duration and nominal duration value *quantifies* duration value set and duration ≤ duration value set
Example:	"366 days" *is less than or equal to* "1 year 1 day"

nominal duration value ***is less than or equal to*** **precise duration value**

Synonymous Form:	nominal duration value ≤ precise duration value
Synonymous Form:	precise duration value *is greater than or equal to* nominal duration value
Synonymous Form:	precise duration value ≥ nominal duration value
Definition:	nominal duration value *quantifies* duration value set and precise duration value *quantifies* duration and duration value set ≤ duration
Example:	"2 years 1 day" *is less than or equal to* "732 days"

precise duration value$_1$ *is less than* precise duration value$_2$

Synonymous Form: precise duration value$_2$ *is greater than* precise duration value$_1$

Synonymous Form: precise duration value$_1$ < precise duration value$_2$

Synonymous Form: precise duration value$_2$ > precise duration value$_1$

Definition: precise duration value$_1$ *quantifies* duration$_1$ and precise duration value$_2$ *quantifies* duration$_2$ and duration$_1$ < duration$_2$

Example: "1 hour 30 minutes" *is less than* "91 minutes"

nominal duration value$_1$ *is less than* nominal duration value$_2$

Synonymous Form: nominal duration value$_2$ *is greater than* nominal duration value$_1$

Synonymous Form: nominal duration value$_1$ < nominal duration value$_2$

Synonymous Form: nominal duration value$_2$ > nominal duration value$_1$

Definition: nominal duration value$_1$ *quantifies* duration value set$_1$ and nominal duration value$_2$ *quantifies* duration value set$_2$ and duration value set$_1$ < duration value set$_2$

Example: "1 month 1 day" *is less than* "1 month 2 days"

precise duration value *is less than* nominal duration value

Synonymous Form: precise duration value < nominal duration value

Synonymous Form: nominal duration value *is greater than* precise duration value

Synonymous Form: nominal duration value > precise duration value

Definition: precise duration value *quantifies* duration and nominal duration value *quantifies* duration value set and duration < duration value set

Example: "366 days" *is less than* "1 year 2 days"

nominal duration value *is less than* precise duration value

Synonymous Form: nominal duration value < precise duration value

Synonymous Form: precise duration value *is greater than* nominal duration value

Synonymous Form: precise duration value > nominal duration value

Definition: nominal duration value *quantifies* duration value set and precise duration value *quantifies* duration and duration value set < duration

Definition: "1 month 1 day" *is less than* "34 days"

11.4 Duration Value Sets

This sub clause defines the concept 'duration value set' and those relationships of that concept that are needed to semantically ground other features of this specification.

duration value set

Definition: set *of* duration values

Possibility: the cardinality *of* a duration valueset *is* 0

Example: the duration value set that *is quantified by* {60 seconds, 64 seconds}

The following concepts support comparison of two duration value sets.

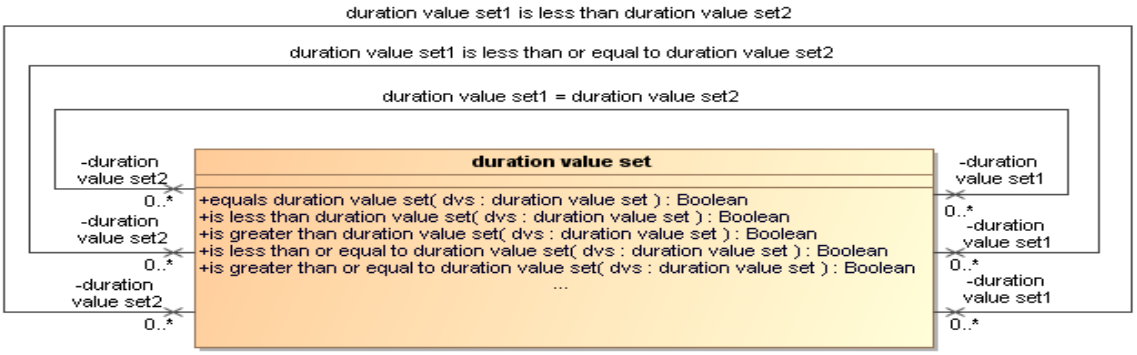

Figure 11.6 - Duration Value Set Comparisons

duration value set$_1$ *equals* duration value set$_2$

Synonymous Form: duration value set$_1$ *is equal to* duration value set$_2$

Synonymous Form: duration value set$_1$ *is equivalent to* duration value set$_2$

Synonymous Form: duration value set$_1$ = duration value set$_2$

Definition: each duration$_1$ *of* duration value set$_1$ = some duration$_2$ *of* duration value set$_2$ and each duration$_2$ *of* duration value set$_2$ = some duration$_1$ *of* duration value set$_1$

Example: the duration value set {1 week, 2 weeks} *equals* the duration value set {7 days, 14 days}

Example: the duration value set {1 day, 2 days} *equals* the duration value set {2 days, 1 day}

duration value set$_1$ *is less than or equal to* duration value set$_2$

Synonymous Form: duration value set$_2$ *is greater than or equal to* duration value set$_1$

Synonymous Form: duration value set$_1$ ≤ duration value set$_2$

Synonymous Form: duration value set$_2$ ≥ duration value set$_1$

Definition: each duration value$_1$ *of* duration value set$_1$ *is less than or equal to* each duration value$_2$ *of* duration value set$_2$

Example: the duration value set {1 day, 2 days} *is less than or equal to* the duration value set {2 days, 4 days}

duration value set$_1$ *is less than* duration value set$_2$

Synonymous Form: duration value set$_2$ *is greater than* duration value set$_1$

Synonymous Form: duration value set$_1$ < duration value set$_2$

Synonymous Form: duration value set$_2$ > duration value set$_1$

Definition: each duration value$_1$ *of* duration value set$_1$ *is less than* each duration value$_2$ *of* duration value set$_2$

Example: the duration value set {1 day, 2 days} *is less than* the duration value set {3 days, 4 days}

Durations can be compared with duration value sets.

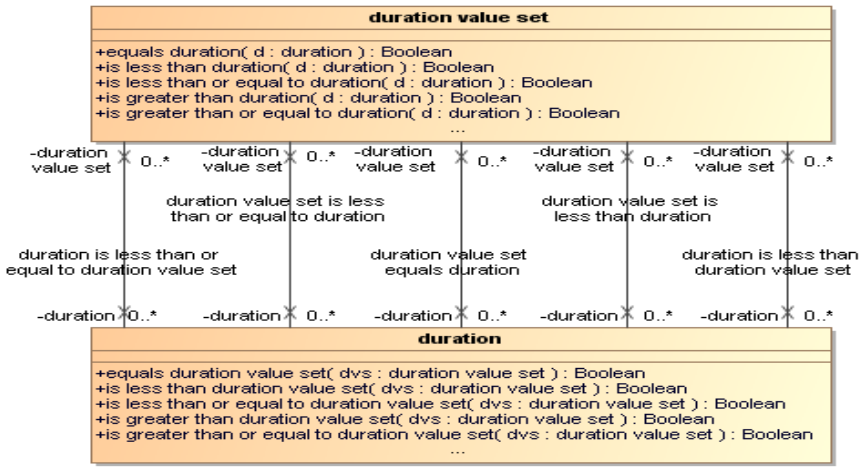

Figure 11.7 - Comparisons among Duration Value Sets and Durations

duration value set *equals* duration

Synonymous Form: duration = duration value set

Synonymous Form: duration value set *equals* duration

Synonymous Form: duration *equals* duration value set

Synonymous Form: duration value set *is equivalent to* duration

Synonymous Form: duration *is equivalent to* duration value set

Definition: each duration value *of* the duration value set *equals* the given duration

Example: the duration value set {1 day} *equals* the duration that *is quantified by* 1 day

duration value set *is less than or equal to* duration

Synonymous Form: duration *is greater than or equal to* duration value set

Synonymous Form: duration value set ≤ duration

Synonymous Form: duration ≥ duration value set

Definition: each duration value *of* the duration value set *is less than or equal to* the given duration

Example: the duration value set {1 day, 2 days} *is less than or equal to* the duration that *is quantified by* 2 days

duration *is less than or equal to* duration value set

Synonymous Form: duration value set *is greater than or equal to* duration

Synonymous Form: duration ≤ duration value set

Synonymous Form: duration value set ≥ duration

Definition: duration *is less than or equal to* each duration value *of* the duration value set

Example: the duration that *is quantified by* 28 days *is less than or equal to* the duration value set {28 days, 29 days}

duration value set *is less than* duration

Synonymous Form: duration *is greater than* duration value set

Synonymous Form: duration value set < duration

Synonymous Form: duration > duration value set

Definition: each duration value *of* the duration value set *is less than* the given duration

Example: the duration value set {1 day, 2 days} *is less than* the duration that *is quantified by* 3 days

duration *is less than* duration value set

Synonymous Form: duration value set *is greater than* duration

Synonymous Form: duration < duration value set

Synonymous Form: duration value set > duration

Definition: duration *is less than* each duration value *of* the duration value set

Example: the duration that *is quantified by* 364 days *is less than* the duration value set {365 days, 366 days}

Specification of compound nominal duration values as duration value sets requires addition and subtraction among durations and duration value sets, and addition and subtraction among two duration value sets.

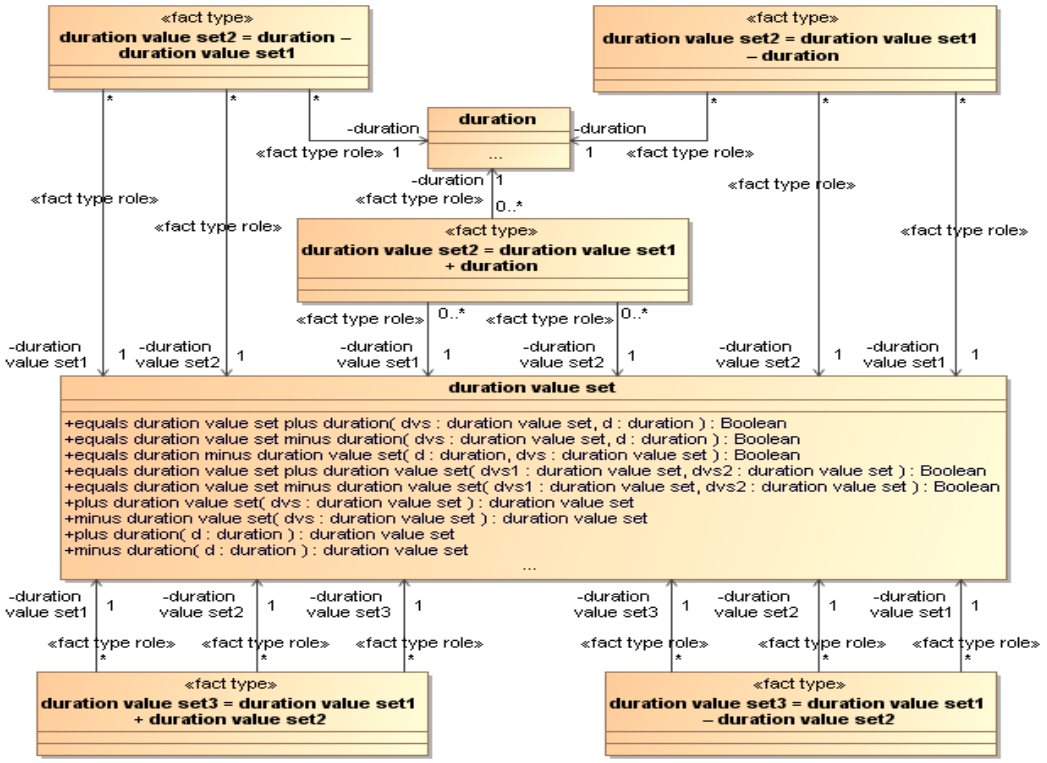

Figure 11.8 - Duration Value Set Arithmetic

duration value set$_2$ *equals* duration value set$_1$ *plus* duration

Synonymous Form: duration value set$_2$ *equals* duration *plus* duration value set$_1$

Synonymous Form: duration value set$_2$ = duration value set$_1$ + duration

Synonymous Form: duration value set$_2$ = duration + duration value set$_1$

Synonymous Form: duration value set$_1$ + duration

Synonymous Form: duration + duration value set$_1$

Definition: each duration value$_2$ *of* the duration value set$_2$ *equals* some duration value$_1$ *of* duration value set$_1$ *plus* the given duration

Example: the duration value set {3 days, 4 days} *equals* the duration that *is quantified by* 2 days *plus* the duration value set {1 day, 2 days}

duration value set$_3$ *equals* duration value set$_1$ *plus* duration value set$_2$

Synonymous Form: duration value set$_3$ = duration value set$_1$ + duration value set$_2$

Definition: each duration value$_3$ *of* duration value set$_3$ = some duration value$_1$ *of* duration value set$_1$ + some duration value$_2$ *of* duration value set$_2$, where the duration value$_1$ and duration value$_2$ are selected to form a Cartesian product of duration value set$_1$ and duration value set$_2$

Note: The result set disregards duplicates. Hence the cardinality of duration value set$_3$ may be less than the product of the cardinalities of duration value set$_1$ and duration value set$_2$.

Example: the duration value set {4 days, 5 days, 6 days} *equals* the duration value set {1 day, 2 days} *plus* the duration value set {3 days, 4 days}

duration value set$_2$ = duration value set$_1$ – duration

Synonymous Form: duration value set$_1$ - duration

Definition: each duration value$_2$ *of* the duration value set$_2$ = some duration value$_1$ *of* duration value set$_1$ – the given duration

Example: the duration value set {2 days, 0 days} = the duration value set {3 days, 1 day} – the duration that *is quantified by* 1 day

duration value set$_2$ = duration – duration value set$_1$

Definition: duration - duration value set$_1$

Synonymous Form: each duration value$_2$ *of* duration value set$_2$ = the given duration - some duration value$_1$ *of* duration value set$_1$

Example: the duration value set {1 day, 0 days} = the duration that *is quantified by* 2 days - the duration value set {1 day, 2 days}

duration value set$_3$ = duration value set$_1$ – duration value set$_2$

Definition: each duration value$_3$ *of* duration value set$_3$ = some duration value$_1$ *of* duration value set$_1$ - some duration value$_2$ *of* duration value set$_2$, where the duration value$_1$ and duration value$_2$ are selected to form a Cartesian product of duration value set$_1$ and duration value set$_2$

Note: The result set disregards duplicates. Hence the cardinality of duration value set$_3$ may be less than the product of the cardinalities of duration value set$_1$ and duration value set$_2$.

Example: the duration value set {-1 days, 0 days, 2 days, 3 days } = the duration value set {3 days, 4 days} - the duration value set {1 days, 4 days}

11.5 Year Values

This sub clause defines the meaning of nominal atomic duration values that use the nominal time unit 'year'. It accounts for the varying numbers of calendar days in Gregorian years, due to leap years, centennial years, and quadricentennial years.

Note: this sub clause defines some concepts, such as 'year remainder', that are only needed to support the concept 'year value *specifies* duration value set'. These supporting concepts need not be explicitly defined in versions of this specification in other modeling systems.

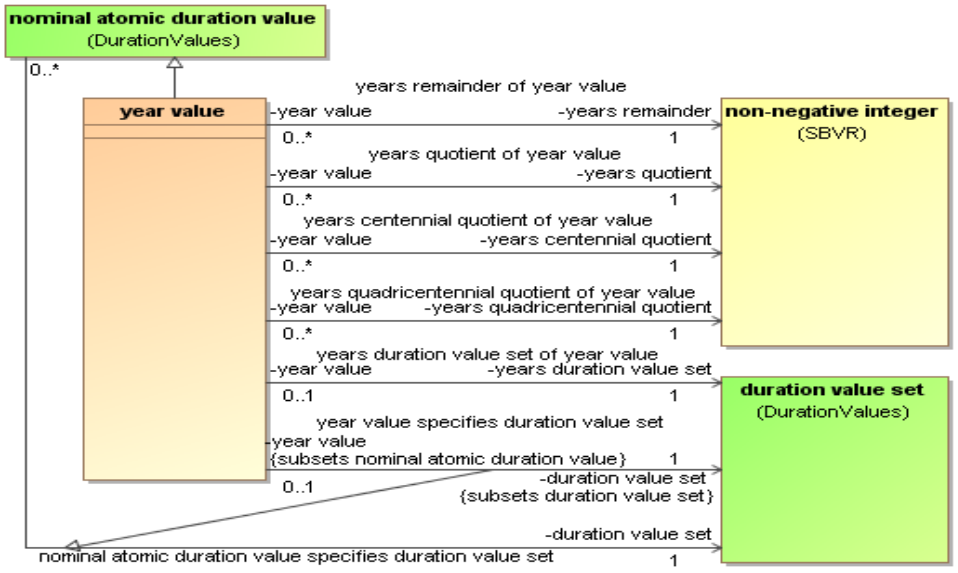

Figure 11.9 - Year Values

year value

Definition: nominal atomic duration value that *has* the time unit 'year'

years remainder

Concept Type: role

General Concept: nonnegative integer

years remainder *of* year value

Definition: the years remainder is the remainder produced by dividing the number *of* the year value by 4

Note: Each 4-year cycle includes exactly 1 leap day.

Example: the years remainder of '5 years' is 1

years quotient

Concept Type: role

General Concept: nonnegative integer

years quotient *of* year value

Definition: the years quotient is the quotient produced by dividing the number *of* the year value by 4

Note: Each 4-year cycle includes exactly 1 leap day.

Example: the years quotient of '11 years' is 2

years centennial quotient

Concept Type: role

General Concept: nonnegative integer

years centennial quotient *of* year value

Definition: the years centennial quotient is the remainder produced by dividing the number *of* the year value by 100

Note: According to [Inter Gravissimas], a leap day is omitted for each centennial year that is not a quadricentennial year.

Example: the years centennial quotient of '5 years' is 0

Example: the years centennial quotient of '301 years' is 3

years quadricentennial quotient

Concept Type: role

General Concept: nonnegative integer

years centennial quotient *of* year value

Definition: the years centennial quotient is the remainder produced by dividing the number *of* the year value by 400

Note: According to [Inter Gravissimas], a leap day is included for each quadricentennial year even though it is a centennial year.

Example: the years quadricentennial quotient of '301 years' is 0

Example: the years quadricentennial quotient of '401 years' is 1

years duration value set

Concept Type: role

Definition: duration value set

years duration value set *of* year value

Definition: the years duration value set is specified by the following table, according to the years remainder *of* the year value:

Note: The duration value sets of the table are constructed so that the first element of each set supports the case where the year value does not include a leap year, and the second member is for the case where the year value does include a leap year.

Table 11.1 - Year Duration Value Sets

year remainder	year duration value set
0	{}
1	{365 days, 366 days}
2	{730 days, 731 days}
3	{1 095 days, 1 096 days}

year value *specifies* duration value set

Definition: the duration value set *is specified by* the formula: (1 day * ((years quotient *of* the year value * 1 461) – (2 *years centennial quotient *of* the year value) + (2 * years quadricentennial quotient *of* the year value)) + years duration set

Note: 1 461 days is the number of days in 4 years, including 1 leap day.

Note: 86 400 seconds is the number of seconds in a day. The formula computes a duration value set of days, taking into account leap days, and converts the result to a duration value set of days.

Note: [Inter Gravissima] specifies that a leap day occurs every 4 years, except every 100 years, with a further exception that a leap day does occur 400 years.

Note: The formula subtracts 2 (rather than 1) leap days for each 100 years because it also subtracts 1 leap day for each 4 years (and a centennial year is also a leap year).

Note: The formula adds 2 (rather than 1) leap days for each 400 years because it also subtracts 2 leap days for each 100 years (and a quadricentennial year is also a centennial year).

11.6 Month values

This sub clause defines the meaning of nominal atomic duration values that use the nominal time unit 'month.' It accounts for the varying numbers of calendar days in the calendar months of the Gregorian calendar. It accounts for leap years by considering that 48 months (4 years of 12 months) includes one leap day (February 29). The computation adjusts for the fact that centennial years have no leap days, but quadricentennial years have one leap day.

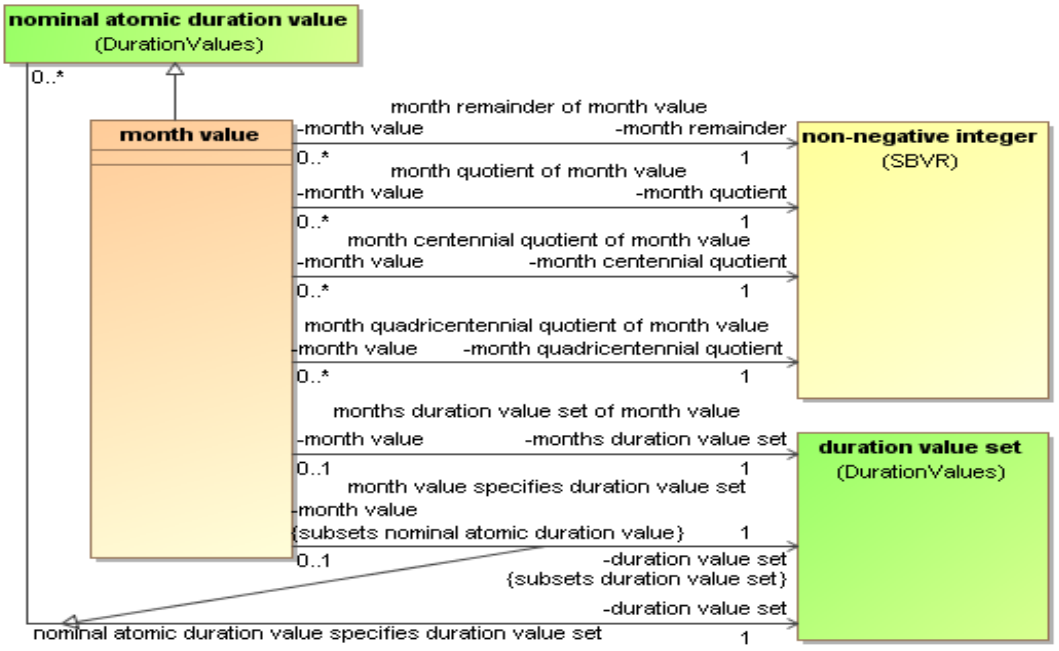

Figure 11.10 - Month Values

Note: this sub clause defines some concepts, such as ‘month remainder’, that are only needed to support the concept ‘month value *specifies* duration value set’. These supporting concepts need not be explicitly defined in versions of this specification in other modeling systems.

month value

Definition: nominal atomic duration value that *has* the time unit ‘month’

months remainder

Concept Type: role

General Concept: nonnegative integer

months remainder *of* month value

Definition: the months remainder is the remainder produced by dividing the number *of* the month value by 48

Note: 48 is the number of months in a 4-year cycle that includes one leap day.

Example: the months remainder of ‘50 months’ is 2

months quotient

Concept Type: role

General Concept: nonnegative integer

months quotient *of* month value

Definition: the months quotient is the quotient produced by dividing the number *of* the month value by 48

Note: 48 is the number of months in a 4-year cycle that includes one leap day.

Example: the months quotient of ‘50 months’ is 1

months centennial quotient

Concept Type: role

General Concept: nonnegative integer

months centennial quotient *of* month value

Definition: the months centennial quotient is the remainder produced by dividing the number *of* the month value by 1 200

Note: 1 200 is 100 years of 12 months. According to [Inter Gravissimas], a leap day is omitted for each centennial year that is not a quadricentennial year.

Example: the months centennial quotient of ‘60 months’ is 0

Example: the months centennial quotient of ‘2405 months’ is 2

months quadricentennial quotient

Concept Type: role

General Concept: nonnegative integer

months centennial quotient *of* month value

Definition: the months centennial quotient is the remainder produced by dividing the number *of* the month value by 4 800

Note: 4 800 is 400 years of 12 months. According to [Inter Gravissimas], a leap day is included for each quadricentennial year even though it is a centennial year.

Example: the months quadricentennial quotient of ‘10 months’ is 0

Example: the months quadricentennial quotient of ‘4 805 months’ is 1

months duration value set

Concept Type: role

Definition: duration value set

months duration value set *of* month value

Definition: the months duration value set is specified by the following table, according to the months remainder *of* the month value

Table 11.2 - Month Duration Value Sets

month remainder (0 – 11)	month duration value set	month remainder (12-23)	month duration value set	month remainder (24-35)	month duration value set	month remainder (36-47)	month duration value set
0	{}	12	{365 days, 366 days }	24	{730 days, 731 days }	36	{1 095 days, 1 096 days }
1	{28 days, 29 days, 30 days, 31 days}	13	{393 days, 394 days, 395 days, 396 days, 397 days}	25	{758 days, 759 days, 760 days, 761 days, 762 days}	37	{1 124 days, 1 125 days, 1 126 days, 1 127 days}
2	{59 days, 60 days, 61 days, 62 days}	14	{424 days, 425 days, 426 days, 427 days, 428 days}	26	{789 days, 790 days, 791 days, 792 days, 793 days}	38	{1 155 days, 1 156 days, 1 157 days, 1 158 days}
3	{89 days, 90 days, 91 days, 92 days}	15	{454 days, 455 days, 456 days, 457 days, 458 days}	27	{819 days, 820 days, 821 days, 822 days, 823 days}	39	{1 185 days, 1 186 days, 1 187 days, 1 188 days}
4	{120 days, 121 days, 122 days, 123 days}	16	{485 days, 486 days, 487 days, 488 days, 489 days}	28	{850 days, 851 days, 852 days, 853 days, 854 days}	40	{1 216 days, 1 217 days, 1 218 days, 1 219 days}
5	{150 days, 151 days, 152 days, 153 days}	17	{515 days, 516 days, 517 days, 518 days, 519 days}	29	{880 days, 881 days, 882 days, 883 days, 884 days}	41	{1 246 days, 1 247 days, 1 248 days, 1 249 days}
6	{181 days, 182 days, 183 days, 184 days}	18	{546 days, 547 days, 548 days, 549 days, 550 days}	30	{911 days, 912 days, 913 days, 914 days, 915 days}	42	{1 277 days, 1 278 days, 1 279 days, 1 280 days}
7	{212 days, 213 days, 214 days, 215 days}	19	{577 days, 578 days, 579 days, 580 days, 581 days}	31	{942 days, 943 days, 944 days, 945 days, 946 days}	43	{1 308 days, 1 309 days, 1 310 days, 1 311 days}

Table 11.2 - Month Duration Value Sets

8	{242 days, 243 days, 244 days, 245 days}	20	{607 days, 608 days, 609 days, 610 days, 611 days}	32	{972 days, 973 days, 974 days, 975 days, 976 days}	44	{1 338 days, 1 339 days, 1 340 days, 1 341 days}
9	{273 days, 274 days, 275 days, 276 days}	21	{638 days, 639 days, 640 days, 641 days, 642 days}	33	{1 003 days, 1 004 days, 1 005 days, 1 006 days, 1 007 days}	45	{1 369 days, 1 370 days, 1 371 days, 1 372 days}
10	{303 days, 304 days, 305 days, 306 days}	22	{668 days, 669 days, 670 days, 671 days, 672 days}	34	{1 033 days, 1 034 days, 1 035 days, 1 036 days, 1 037 days}	46	{1 399 days, 1 400 days, 1 401 days, 1 402 days}
11	{334 days, 335 days, 336 days, 337 days}	23	{699 days, 700 days, 701 days, 702 days, 703 days}	35	{1 064 days, 1 065 days, 1 066 days, 1 067 days, 1 068 days}	47	{1 430 days, 1 431 days, 1 432 days, 1 433 days}

month value *specifies* duration value set

Definition: the duration value set *is specified by* the formula: (1 day * ((months quotient *of* the month value * 1 461) – (2 * months centennial quotient *of* the month value) + (2 * months quadricentennial quotient *of* the month value))) + months duration set

Note: 1 461 days is the number of days in 48 months, including 1 leap day.

Note: 86 400 seconds is the number of seconds in a day. The formula computes a duration value set of days, taking into account leap days, and converts the result to a duration value set of days.

Note: [Inter Gravissima] specifies that a leap day occurs every 4 years, except every 100 years, with a further exception that a leap day does occur 400 years.

Note: The formula subtracts 2 (rather than 1) leap days for each 100 years because it also subtracts 1 leap day for each 4 years (and a centennial year is also a leap year).

Note: The formula adds 2 (rather than 1) leap days for each 400 years because it also subtracts 2 leap days for each 100 years (and a quadricentennial year is also a centennial year).

12 Time Coordinates and Time Scale Conversions (normative)

Time coordinates are locations in time stated in terms of one or more time scales. Most time coordinates are compound, meaning they have multiple components. Examples are "July 1, 2010 12:43:55", "week 41 day 6", and "1999 day 45". This clause specifies which combinations of atomic time coordinates form legitimate compound time coordinates. Invalid combinations typically omit intermediate time units. For example, "2011 12:43:55" makes no sense.

This specification does NOT specify how time coordinates are externally represented, for example on a monitor or in printed form. Many different external formats are employed among different languages and cultures. Representation formats are the choice of individual tools.

Time coordinates *indicate* time points, meaning that each time coordinate identifies one or more time points on a time scale. When a time coordinate incorporates a year number, the time coordinate *indicates* a single time point and is called an 'absolute time coordinate'. For example, "January 3, 2011" indicates a particular calendar day. When a time coordinate omits the year, the time coordinate *indicates* a set of time points and is called a 'relative time coordinate'. For example, "January 3" *indicates* a set of calendar days, one for every calendar year.

When more than one time coordinate *indicates* the same time point, they are said to be *equivalent*. For example, "January 3, 2011" *is equivalent to* "2011 day 3" because the two time coordinates *indicate* the same calendar day. Determining equivalence is not easy because of the incorporation of leap days in some calendar years. For example, whether the 182nd day of the calendar year is before or the same as July 1 of the same calendar year depends upon whether the calendar year is a leap year.

This clause introduces the concept 'time set' to help answer such questions. A time set represents a choice of one or more time periods on a single time scale. The concept models the idea that a time coordinate may *indicate* a choice of several time periods. For example, February may be either the 32nd through the 59th calendar day of a calendar year, or the 32nd through the 60th calendar day of a calendar year. Time sets make possible the comparison of time coordinates such as "July 13, 2011" and "2011 week 28 day 3" by converting both time coordinates to the Gregorian days scale.

12.1 Time Coordinates

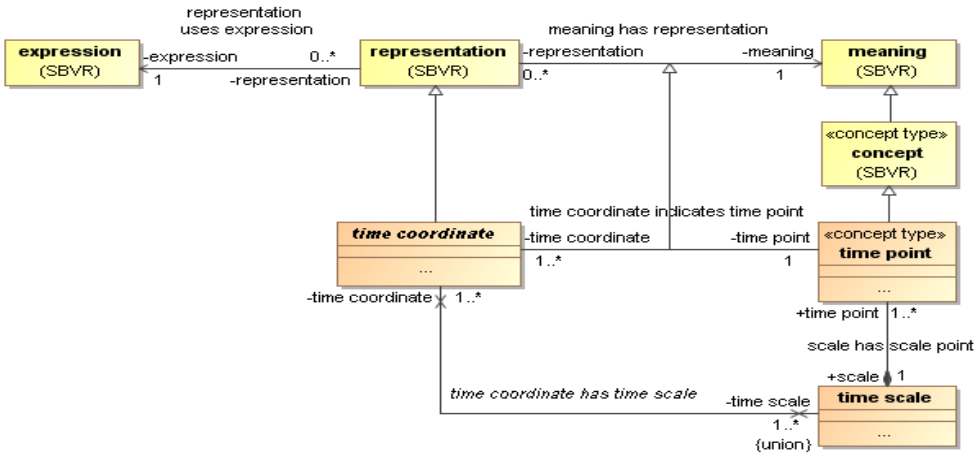

Figure 12.1 - Time Coordinate

time coordinate

Synonym:	time stamp
Definition:	representation *of* a time point by an atomic time coordinate or compound time coordinate that *indicates* the time point
Reference Scheme:	the indices and time scales *of* the time coordinate
Example:	January 2009, 2009 month 1
Note:	Time coordinates may be either atomic or compound (see sub clause 12.1.2).
Note:	Time coordinates may be given by an index and an implied time scale (e.g.,"month 1"), or by a name (e.g., "January").
Note:	Named time coordinates, such as 'Tuesday,' are defined in terms of an index and a time scale. Per SBVR, named time coordinates are individual concepts because their extension contains one thing: a particular time point on a particular time scale.
Note:	Particular kinds of time coordinates are defined below, in sub clauses 12.3.1, 12.3.2, and 12.3.3.

time coordinate *indicates* time point

Definition:	time point *is* the scale point *of* the time scales *of* the time coordinate that *has* the indices *of* the time coordinate
Necessity:	Each time point *is indicated by* at least one time coordinate.
Necessity:	Each time coordinate *indicates* exactly one time point.
Possibility:	A time point *is indicated by* more than one time coordinate.
Note:	See 'atomic time coordinate *indicates* time point' and 'compound time coordinate *indicates* time point' for definitions of exactly how a time coordinate *indicates* a time point.

12.1.1 Absolute and Relative Time Coordinates

It is convenient to distinguish between absolute time coordinates (time coordinates that include a calendar year and hence can be located on the Time Axis) and relative time coordinates (time coordinates that are relative to some larger time unit).

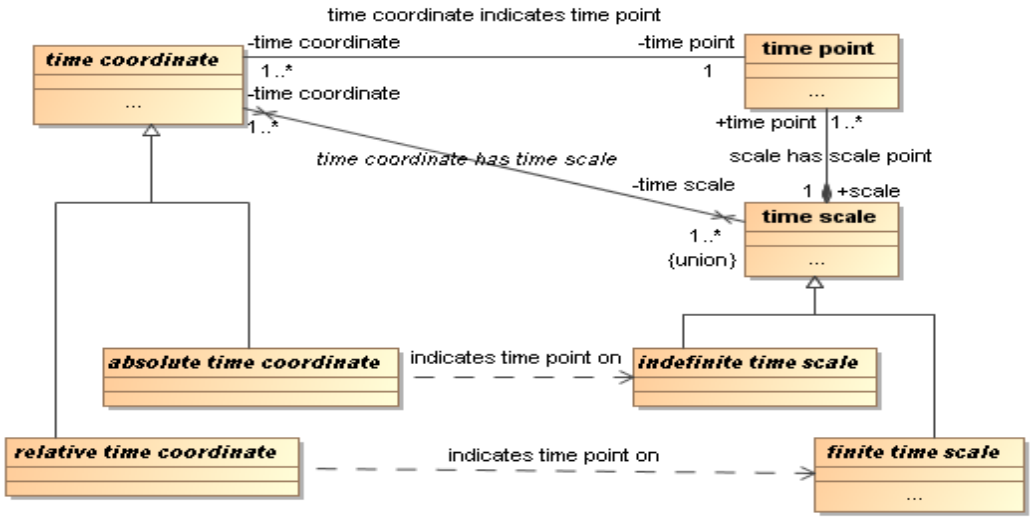

Figure 12.2 - Absolute and Relative Time Coordinates

absolute time coordinate

Definition: time coordinate that *has* an indefinite time scale and *refers to* exactly one time interval

Example: 12 November 1969

relative time coordinate

Definition: time coordinate that *does* not *have* an indefinite time scale and *refers to* more than one time interval

Note: A relative time coordinate *indicates* a time point that is "within" a larger time point (e.g., an hour of day is part of a calendar day), and thus recurs in each instance of the larger time point.

Example: 12 November (which recurs every calendar year)

12.1.2 Atomic and Compound Time Coordinates

As with duration values, time coordinates can be atomic (reference just one time scale, as in "5 p.m.") or compound (referencing multiple time scales, as in "5:00 p.m.", which combines an hour-of-day and a minute-of-hour).

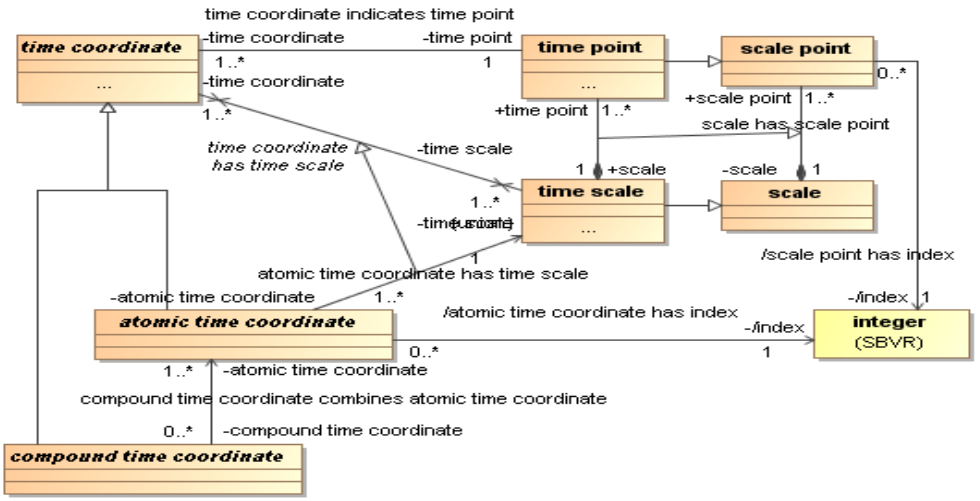

Figure 12.3 - Atomic and Compound Time Coordinates

time expression

Definition: expression that *is of* exactly one index *on* exactly one time scale

Note: A time expression may be specified as an index and a time scale (e.g.,"month 1") or as an individual concept (e.g., "January") that is defined in terms of an integer and a time scale. The time scale may be implicit (e.g., "2010" used as a time coordinate implies the Gregorian years scale).

Example: "month 1" expresses index 1 on some time scale that has granularity 'month'

time expression *has* index

Definition: The index is assigned to a time point *of* the time scale *of* the time expression.

Necessity: Each time expression *has* exactly one index.

Note: "Index" is defined in Annex D.1 as a role of an integer.

Note: The assignment of an index to a time point depends upon the index origin value of the time scale. Hours of day, minutes of hour, and seconds of minute are conventionally numbered from 0, while calendar months, calendar weeks, and calendar days are numbered from 1. Calendar years are numbered from 1 in the Gregorian calendar, but are numbered from 0 by [ISO 8601]. See 'time index.'

Example: The index '10' of the atomic time coordinate 'day 10'.

time expression *has* time scale

Definition: The time expression *indicates* exactly one time point *of* the time scale.

Necessity: Each time expressions *has* exactly one time scale.

Example: The time scale 'Gregorian years scale' that is meant by the atomic time coordinate '2010'.

atomic time coordinate

Definition: time coordinate that *represents* a time point by exactly one time expression

Reference Scheme: The index *of* the time expression and the time scale *of* the time expression.

Example: 2010, day 53, Tuesday

atomic time coordinate *indicates* time point

Synonymous Form: time point *indicated by* atomic time coordinate

Definition: the index *of* the time expression *of* the atomic time coordinate *is* the index *of* the time point *on* the time scale *of* the time point

Note: Each kind of atomic time coordinate defines the kind of time point that it *indicates*.

Example: "week 15", indicating the 15th week of year, which is on the year of weeks scale.

Note: Some time points are given by individual concepts that are defined in terms of indices and time points.

Example: "February" *indicates* the time point that is Gregorian month of year 2 *on* the Gregorian year of months scale

compound time coordinate

Definition: time coordinate that *represents* a time point by more than one time expression

Example: "2010" is a precise atomic time coordinate indicating a Gregorian year on the Gregorian years scale.

Example: "January 2010" represents 'January' on the Gregorian year of months scale, and '2010' on the Gregorian years scale, combined to *indicate* a particular Gregorian month on the Gregorian months scale.

Example: "'1 February' is the first day of February" mentions (rather than uses) "1 February". The mention means 'February' on the Gregorian year of months scale, 'day 1' on the Gregorian month of days scale, combined to *indicate* Gregorian day 32 on the Gregorian year of days scale.

Example: "'1 March' is the first day of March" mentions (rather than uses) "1 March". The mention means 'March' on the Gregorian year of months scale, 'day 1' on the Gregorian month of days scale, combined to *indicate* the time set {Gregorian day 60, Gregorian day 61} on the Gregorian year of days scale. The time set models the idea that the meaning of "1 March" depends upon whether it is a common year or a leap year.

Example: "Tax returns are due each 15 April." The quantifier and the use (rather than mention) of "15 April" mean a set of Gregorian days, one in each Gregorian year.

compound time coordinate *has* atomic time coordinate

Definition: The atomic time coordinate *has* a time expression *of* the compound time coordinate.

Necessity: Each time expression *of* the compound time coordinate *is* an expression *of* an atomic time coordinate.

compound time coordinate *indicates* time point

Synonymous Form: time point *indicated by* compound time coordinate

Definition: The atomic time coordinates *combine* to *indicate* the time point.

Possibility: A time point *is indicated by* more than one compound time coordinate.

Note: See sub clause 12.3 for details about how time expressions *are combined*.

Example: "January 4, 2010" *indicates* the time point that *is indicated by* "2010 day 4"

12.1.3 Time Coordinate Equivalence

Equivalence of time coordinates captures the idea that they can mean the same thing though given differently. For example, "February 15" and "day 46" *are equivalent*.

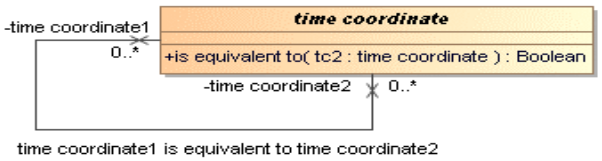

Figure 12.4 - Time Coordinate Equivalence

time coordinate$_1$ *is equivalent to* time coordinate$_2$

Definition: time coordinate$_1$ *indicates* a time point$_1$ and time coordinate$_2$ *indicates* a time point$_2$ and time point$_1$ *is* time point$_2$

Example: "2010 day 3" *is equivalent to* "January 3, 2010"

Example: "March" *is equivalent to* "month 3"

Example: "March 1" refers to the set {Gregorian day of year 60 in common years, Gregorian day of year 61 in leap years}. Therefore March 1 is not equivalent to Gregorian day of year 61.

12.2 Time sets

A 'time set' represents a choice of one or more possible time point sequences on a time scale. Each time point sequence may contain one or multiple time points. This concept models the idea that a relative time coordinate may *convert to* a choice of several time point sequences. In particular, the following kinds of time points *convert to* time sets on the Gregorian year of day scale depending upon whether the year is a common year or a leap year, or upon which calendar day is the first week day of the first week of year of the Gregorian year:

every Gregorian month that is after January
every week of year
every weekday of year

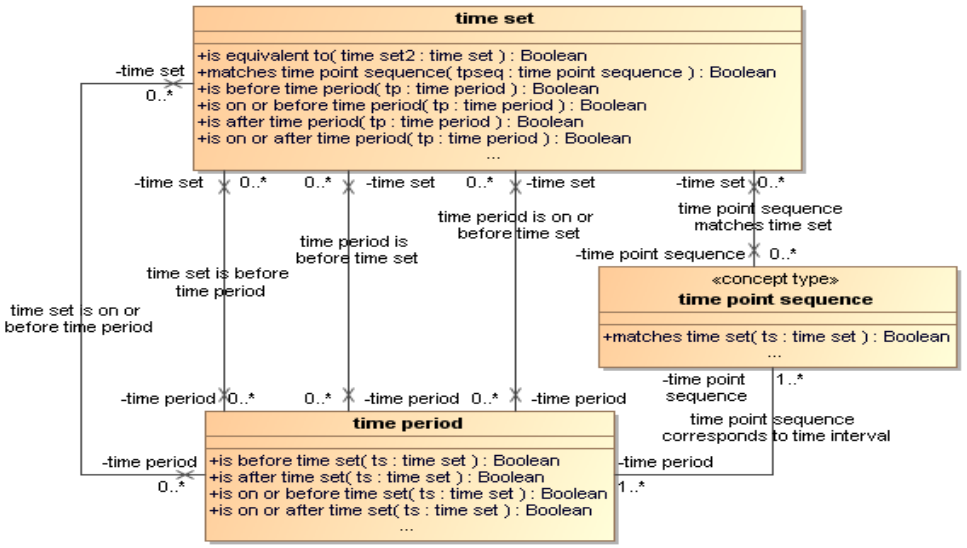

Figure 12.5 - Time Sets

time set

Definition: set *of* time point sequences

Necessity: the cardinality *of* a time set *is greater than* 0

Necessity: the time scale$_1$ *of* each time point sequence$_1$ *of* the time set *is* the time scale$_2$ *of* each time point sequence$_2$ *of* the time set

Example: the time set {Gregorian day of year 59, Gregorian day of year 60}

time set$_2$ *is* duration value *after* time set$_1$

Synonymous Form: duration value *after* time set$_1$ *is* time set$_2$

Synonymous Form: duration value *after* time set$_1$

Synonymous Form: time set$_2$ *equals* duration value *plus* time set$_1$

Synonymous Form: time set$_2$ = time set$_1$ + duration value

Synonymous Form: time set$_2$ = duration value + time set$_1$

Synonymous Form: time set$_1$ *plus* duration value

Synonymous Form: duration value *plus* time set$_1$

Synonymous Form: time set$_1$ + duration value

Synonymous Form: duration value + time set$_1$

Definition: each time point sequence *of* time set$_2$ *is* the duration that *is quantified by* duration value *after* a time point sequence *of* time set$_1$

Necessity: The granularity *of* the time scale *of* each time point sequence *of* time set$_1$ *is* the time unit *of* duration value.

Example: {Gregorian day of year 59, Gregorian day of year 60} *is* 1 day *after* {Gregorian day of year 58, Gregorian day of year 59}

Example: {Gregorian day of year 100 *through* Gregorian day of year 110} *is* 1 day *after* {Gregorian day of year 99 *through* Gregorian day of year 109}

time set$_2$ *is* duration value *before* time set$_1$

Synonymous Form: duration value *before* time set$_1$ *is* time set$_2$

Synonymous Form: duration value *before* time set$_1$

Synonymous Form: time set$_2$ = time set$_1$ - duration

Synonymous Form: time set$_1$ *minus* duration

Synonymous Form: time set$_1$ - duration

Definition: each time point sequence *of* time set$_2$ *is* duration value *before* a time point sequence *of* time set$_1$

Necessity: The granularity *of* the time scale *of* each time point sequence *of* time set$_1$ *is* the time unit *of* duration value.

Example: {Gregorian day of year 59, Gregorian day of year 60} *is* 1 day *before* {Gregorian day of year 60, Gregorian day of year 61}

Example: {Gregorian day of year 49 *through* Gregorian day of year 80} *is* 1 day *before* {Gregorian day of year 50 *through* Gregorian day of year 81}

time set$_1$ *is equivalent to* time set$_2$

Definition: each time point sequence$_1$ *of* time set$_1$ *is* some time point sequence$_2$ *of* time set$_2$ and each time point sequence$_2$ *of* time set$_2$ *is* some time point sequence$_1$ *of* time set$_1$

Example: {Gregorian day of year 59 *through* Gregorian day of year 60, Gregorian day of year 60 *through* Gregorian day of year 61} *is equivalent to* {Gregorian day of year 60 *through* Gregorian day of year 61, Gregorian day of year 59 *through* Gregorian day of year 60}

time point sequence *matches* time set

Synonymous Form: time set *matches* time period

Definition: time point sequence *is* some time point sequence *of* time set

Example: Gregorian day of year 60 *matches* March 1 because March 1 is either Gregorian day of year 60 or Gregorian day of year 61

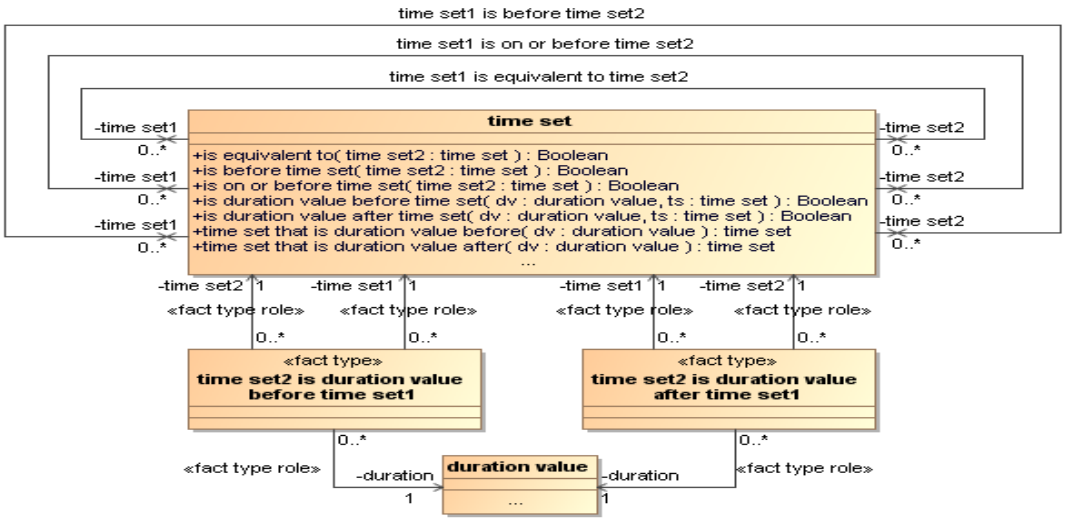

Figure 12.6 - Time Set Relations

time set$_1$ *is on or before* time set$_2$

Synonymous Form: time set$_2$ *is on or after* time set$_1$

Synonymous Form: time set$_1$ ≤ time set$_2$

Synonymous Form: time set$_2$ ≥ time set$_1$

Definition: each time point sequence$_1$ *of* time set$_1$ *corresponds to* a time interval$_1$ that *is before* the time interval$_2$ that *instantiates* each time point sequence$_2$ *of* time set$_2$

Example: {Gregorian day of year 100 *through* Gregorian day of year 101} *is on or before* {Gregorian day of year 101 *through* Gregorian day of year 102}

time period *is on or before* time set

Synonymous Form: time set *is on or after* time period

Synonymous Form: time period ≤ time set

Synonymous Form: time set ≥ time period

Definition: time period *is before* the time interval that *instantiates* each time point set *of* time set

Example: Gregorian day of year 102 *is on or before* {Gregorian day of year 102, Gregorian day of year 103}

Example: “January”*is on or before* {Gregorian day of year 102 *through* Gregorian day of year 103}

time set *is on or before* time period

Synonymous Form: time period *is on or after* time set

Synonymous Form: time set ≤ time period

Synonymous Form: time period ≥ time set

Definition: the time interval that *instantiates* each time point sequence *of* time set *is before* time period

Example: {Gregorian day of year 102, Gregorian day of year 103} *is on or before* Gregorian day of year 103

time set$_1$ *is before* time set$_2$

Synonymous Form: time set$_2$ *is after* time set$_1$

Synonymous Form: time set$_1$ < time set$_2$

Synonymous Form: time set$_2$ > time set$_1$

Definition: the time interval$_1$ that *instantiates* each time point sequence$_1$ *of* time set$_1$ < the time interval$_2$ that *instantiates* each time period$_2$ *of* time set$_2$

Example: {Gregorian day of year 100 *through* Gregorian day of year 101} *is before* {Gregorian day of year 102 *through* Gregorian day of year 103}

time period *is before* time set

Synonymous Form: time set *is after* time period

Synonymous Form: time period < time set

Synonymous Form: time set > time period

Definition: time period *precedes* the time interval that *instantiates* each time point sequence *of* time set

Example: Gregorian day of year 101 *is before* {Gregorian day of year 102 *through* Gregorian day of year 103}

time set *is before* time period

Synonymous Form: time period *is after* time set

Synonymous Form: time set < time period

Synonymous Form: time period > time set

Definition: the time interval that *instantiates* each time point sequence *of* time set *precedes* time period

Example: {Gregorian day of year 102 *through* Gregorian day of year 103} *is before* Gregorian day of year 104

12.3 Standard Time Coordinates

The meaning of every time coordinate is defined with respect to a particular time scale. For example, year time coordinates are defined on the Gregorian year scale. Commonly-used time coordinates are specified earlier in this document. Less commonly-used time coordinates are defined here.

A time coordinate can be absolute or relative, and atomic or compound. This yields four combinations.

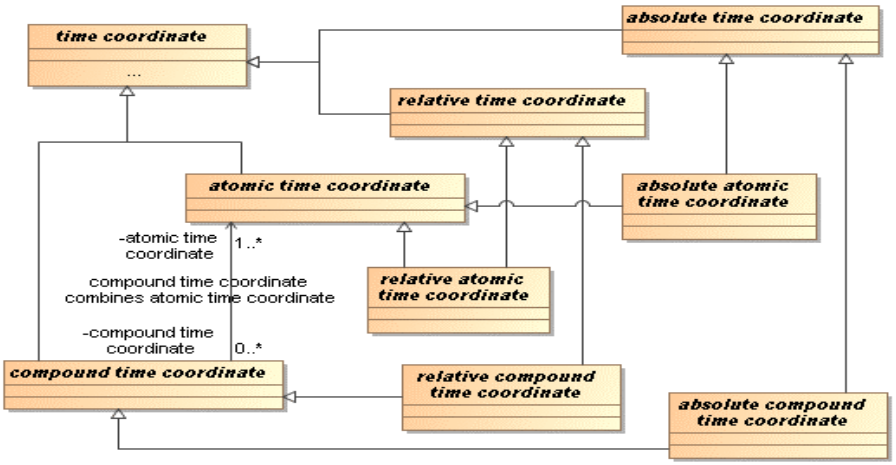

Figure 12.7 - Time Coordinate Types

absolute atomic time coordinate

Definition: absolute time coordinate that *is* an atomic time coordinate

Example: 2010

absolute compound time coordinate

Definition: absolute time coordinate that *is* a compound time coordinate

Example: 5 April 2010

relative atomic time coordinate

Definition: relative time coordinate that *is* an atomic time coordinate

Example: January

relative compound time coordinate

Definition: relative time coordinate that *is* a compound time coordinate

Example: 10:00

Compound time coordinates are defined using the *combines* verb concept, which specifies the combination of atomic time coordinates that may be used together to compose a compound time coordinate.

compound time coordinate *combines* atomic time coordinate

Synonymous Form: atomic time coordinate *of* coordinate time coordinate

Definition: the atomic time coordinates jointly specify the compound time coordinate

Note: The formula used to combine the indices of the atomic time coordinates is specific to each kind of compound time coordinate. Such formulae typically use "index *selects* time point *on* time scale" to specify exactly how the atomic time coordinates relate to time scales, and "finite time scale *subdivides* time point" to relate these time scales to each other.

Necessity: The cardinality *of* the set *is greater than* 1.

Necessity: Each atomic time coordinate$_1$ *of* the compound time coordinate *is* not an atomic time coordinate$_2$ *of* the compound time coordinate.

Necessity: Each atomic time coordinate$_1$ *of* the compound time coordinate *has* a time scale that *is* not *equal to* each atomic time coordinate$_2$ *of* the compound time coordinate.

Example: "2010 month 3" *combines* {2010, month 3} *to select* 3 *on* the Gregorian year-of-months scale

Calendar scales (i.e., those involving calendar years, calendar months, calendar weeks, and calendar days) use index origin value 1, while time-of-day scales (i.e., those involving hours of day, minutes of hour and seconds of minute) use index origin value 0. In all cases other than the Gregorian years scale, the index original element is the first element of the time scale. For clarity, the index origin value and index origin element of each time scale is included with the time scale.

12.3.1 Gregorian Calendar Time Coordinates

This sub clause defines several Gregorian time coordinates and their meaning in terms of time scales. It also "anchors" the Gregorian calendar on the Time Axis per the signing of the Convention du Mètre.

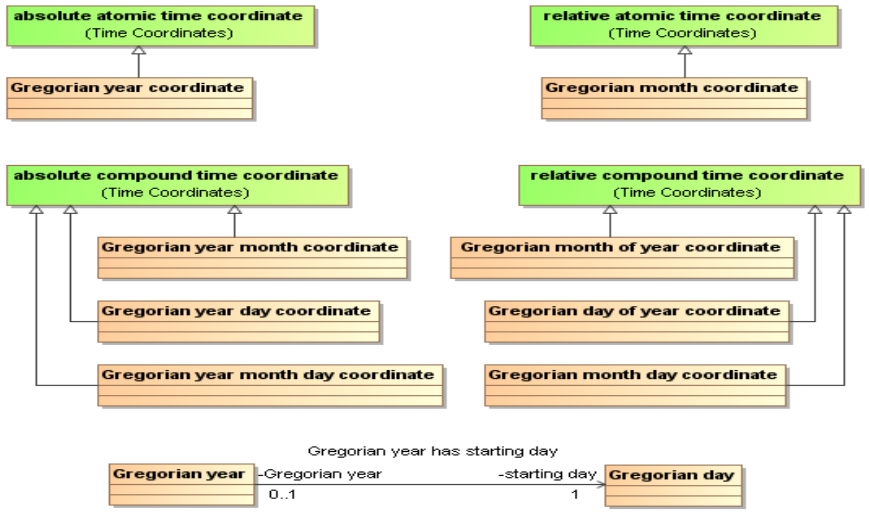

Figure 12.8 - Gregorian Calendar Time Coordinates

- Gregorian year coordinate composed of a Gregorian year, for example "2010"
- Gregorian month coordinate composed of a Gregorian month, for example "January"
- Gregorian day of year coordinate composed of a Gregorian day of year, for example "Gregorian day of year 360"
- Gregorian day of month coordinate composed of a Gregorian day of month, for example "Gregorian day of month 14"
- Gregorian year month coordinate composed of a Gregorian year and a Gregorian month of year, for example "July 2010"
- Gregorian year month day coordinate composed of a Gregorian year, a Gregorian month of year, and a Gregorian day of month, for example "9 July 2010"
- Gregorian year day coordinate composed of a Gregorian year and a Gregorian day of year, for example "2010 day 33"

- Gregorian month day coordinate composed of a Gregorian month of year and a Gregorian day of month, for example "9 July"

Gregorian year coordinate

Definition: absolute atomic time coordinate that *indicates* a Gregorian year that *has* the index equal to the index *of* the Gregorian year coordinate

Example: 2010

Gregorian month coordinate

Definition: relative atomic time coordinate that *indicates* a Gregorian month of year that *has* the index equal to the index *of* the Gregorian month coordinate

Necessity: Each Gregorian month coordinate *is greater than or equal to* 1.

Necessity: Each Gregorian month coordinate *is less than or equal to* 12.

Example: "January" and "month 1" *indicate* the same Gregorian month of year

Gregorian day of year coordinate

Definition: relative atomic time coordinate that *indicates* a Gregorian day of year that *has* the index equal to the index *of* the Gregorian day of year coordinate

Necessity: Each Gregorian day of year coordinate *is greater than or equal to* 1.

Necessity: Each Gregorian day of year coordinate *is less than or equal to* 366.

Example: "day 45" and "14 February" *indicate* the same Gregorian day

Gregorian day of month coordinate

Definition: relative atomic time coordinate that *indicates* a Gregorian day of month that *has* the index equal to the index *of* the Gregorian day of month coordinate

Necessity: Each Gregorian day of month coordinate *is greater than or equal to* 1.

Necessity: Each Gregorian day of month coordinate *is less than or equal to* 31.

Example: "14 February" *indicate* the Gregorian day of month that has index 14

These absolute compound time coordinates support various combinations of Gregorian years, Gregorian month of years, and calendar days.

Gregorian year month coordinate

Definition: absolute compound time coordinate that *combines* the set *of* {a Gregorian year coordinate, a Gregorian month coordinate} and *indicates* the Gregorian month that *has* index 12 times (the index *of* the Gregorian year coordinate minus 1) plus (the index *of* the Gregorian month coordinate minus 1)

Note: The definition subtracts 1 from the indices of the Gregorian year coordinate and Gregorian month coordinate because these are index origin value 1.

Example: "2010 month 3" *combines* the set *of* {2010, month 3}, and *indicates* the Gregorian month that *has* index 24 123

starting day

Concept Type: role

Definition: Gregorian day that is the first calendar day of some Gregorian year

Gregorian year *has* starting day

Definition: the index *of* starting day equals 685 263 plus 365 times (index *of* Gregorian year minus 1601) plus ((index *of* Gregorian year minus 1601) divided by 4) minus (((index *of* Gregorian year minus 1601) divided by 100) multiplied by 2) plus (((index *of* Gregorian year minus 1601) divided by 400) multiplied by 2)

Necessity: The index *of* Gregorian year *is greater than* 1600.

Note: In mathematical form, the definition above is:

sd = 685 263 + (365 * y) + (y/4) - ((y/100)*2)
+ ((y/400) * 2)

where:

sd is the index of the starting day
y is the index of a Gregorian year – 1601
y >= zero
/ is integer division

Note: 685 263 is the index of 1 January 1601, computed as 684 609 (index of 15 October 1582) plus 6 654 days from 15 October 1582 through 1 January 1601.

Note: 1 January 1601 is used as the basis for this formula because the pattern of leap days is consistent since 1601. It is the first day after the first quadricentennial year after [Inter Gravissimas]. This day is picked because the first day of a Gregorian year does not include any leap day that occurs during that Gregorian year.

Note: The definition compensates for leap days by adding 1 for each 4^{th} year, subtracting 2 for each 100^{th} year (because 1 is already added for every 4^{th}), and adding 2 for each 400^{th} year (because 2 is already subtracted for every 100^{th}), per [Inter Gravissimas].

Note: This formula is valid only for Gregorian calendar years after 1600.

Example: The first calendar day of 2010 is Gregorian day 830 991.

Gregorian year month day coordinate

Definition: absolute compound time coordinate that *combines* the set *of* {a Gregorian year coordinate, a Gregorian month coordinate, a Gregorian day of month coordinate} and *indicates* the Gregorian day that *has* starting day *of* the Gregorian year *indicated by* the Gregorian year coordinate, plus the value taken from the table of calendar days at the start of each month (below) as indexed by the index of the Gregorian month coordinate and whether the Gregorian year coordinate *indicates* a leap year, plus the index of the Gregorian day of month coordinate

Example: "2010 month 3 day 15" *combines* the set *of* {2010, month 3, day 15}, and *indicates* the Gregorian day that *has* index 683 768 plus 49 275 (number of calendar days from the start of 1875 to the start of 2010 ignoring leap days) plus 33 leap days minus 2 leap days in centennial years plus 1 leap day in quadricentennial years plus 59 (from the table) plus 15 to give index 733 149.

Table 12.1 - Index of the First Gregorian Day of Year of Each Gregorian Month of Year

Gregorian month by month number	Gregorian month by name	Gregorian day of year from the start of the Gregorian year to this Gregorian month in common years	Gregorian day of year from the start of the Gregorian year to this Gregorian month in leap years
1	January	1	1
2	February	32	33
3	March	60	61
4	April	91	92
5	May	121	122
6	June	152	153
7	July	182	183
8	August	213	214
9	September	244	245
10	October	274	275
11	November	305	306
12	December	335	336

The table shown above is derived from Table 1 of [ISO 8601].

Gregorian year day coordinate

Definition: absolute compound time coordinate that *combines* the set *of* {a Gregorian year coordinate, a Gregorian day of year coordinate} and *indicates* the Gregorian day that *has* starting day *of* the Gregorian year *indicated by* the Gregorian year coordinate, plus the index of the Gregorian day of year coordinate

Example: "2010 day 45" *combines* the set *of* {2010, day 45}, and *indicates* the Gregorian day that *has* index 683 768 plus 49 275 (number of calendar days from the start of 1875 to the start of 2010 ignoring leap days) plus 33 leap days minus 2 leap days in centennial years plus 1 leap day in quadricentennial years plus 45 to give index 733 120.

Gregorian month day coordinate

Definition: relative compound time coordinate that *combines* the set *of* { a Gregorian month coordinate, a Gregorian day of month coordinate }, and *indicates* the time set {Gregorian day of year from the start of the calendar year to the calendar month that has the index of the Gregorian month coordinate in common years, Gregorian day of year from the start of the calendar year to the calendar month that has the index of the Gregorian month coordinate in leap years } *plus* the index *of* the Gregorian day of month coordinate *minus* 1 day

Note: A Gregorian month day coordinate does not include a year number, so there is no way to know whether a February date is of a 28-day or 29-day February. For this reason, every Gregorian month coordinate after February 28 *indicates* either of two possible days of year, one for the case of a common year, and one for a leap year.

Note: The definition subtracts 1 from the sum of the starting day number of the calendar month and the index of the Gregorian day of month coordinate because calendar months use index origin value 1.

Example: "15 June" *combines* the set *of* {June, day 15}, and *indicates* the time set {165 calendar days, 166 calendar days}

Example: "15 January" *combines* the set *of* {January, day 15}, and *indicates* the time set {15 calendar days}

12.3.2 Time of Day Coordinates

This sub clause defines the following relative time coordinates and time scales for these combinations of time of day time units:

- hour coordinate, composed of an hour of day, for example "hour 10" or "10 a.m."
- minute coordinate, composed of a minute of hour, for example "minute 33"
- second coordinate, composed of a second of minute, for example "second 27"
- hour minute coordinate, composed of an hour of day and a minute of hour, for example "10:33"
- hour minute second coordinate, composed of an hour of day, a minute of hour, and a second of minute, for example "10:33:27"

This specification does not define time coordinates and time scales for fractions of seconds (e.g., milliseconds). Business vocabularies may extend this specification as needed to address fractional seconds.

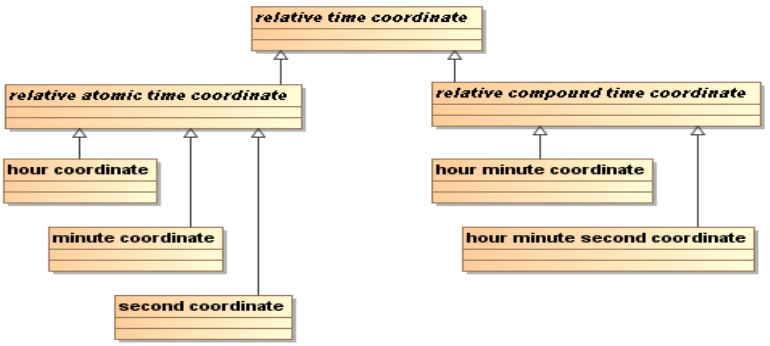

Figure 12.9 - Time of Day Coordinates

hour coordinate

Definition: relative atomic time coordinate that *indicates* the hour of day that *has* the index equal to the index *of* the hour coordinate

Necessity: Each hour coordinate *is greater than or equal to* 0.

Necessity: Each hour coordinate *is less than or equal to* 23.

Example: "11 p.m." and "23:00" *indicate* the same hour of day

minute coordinate

Definition: relative atomic time coordinate that *indicates* the minute-of-hour that *has* the index equal to the index *of* the minute coordinate

Necessity: Each minute coordinate *is greater than or equal to* 0.

Necessity: Each minute coordinate *is less than or equal to* 59.

Example: minute 23

Note: This type of time coordinate is not common in everyday use, but is defined here to support the concepts 'hour minute coordinate' and 'hour minute second coordinate'

second coordinate

Definition: relative atomic time coordinate that *indicates* a second of minute that *has* the index equal to the index *of* the second coordinate

Necessity: Each second coordinate *is greater than or equal to* 0.

Necessity: Each second coordinate *is less than or equal to* 59.

Example: second 45

Note: This type of time coordinate is not common in everyday use, but is defined here to support the concept 'hour minute second coordinate'

hour minute coordinate

Definition: relative compound time coordinate that *combines* the set *of* {an hour coordinate, a minute coordinate} and *indicates* the minute of day that *has* index 60 times the index *of* the hour coordinate plus the index *of* the minute coordinate

Example: "11:23 a.m." *combines* the set *of* {11 a.m, minute 23}, and *indicates* the minute of day that *has* index 683

hour minute second coordinate

Definition: relative compound time coordinate that *combines* the set *of* {an hour coordinate, a minute coordinate, a second coordinate}, and *indicates* the second of day that *has* index 3 600 times the index *of* the hour coordinate plus 60 times the index *of* the minute coordinate plus the index *of* the second coordinate

Example: "11:23:49 a.m." *combines* the set *of* {11 a.m, minute 23, second 49}, and *indicates* the second of day that *has* index 36 432

12.3.3 Week Time Coordinates

This sub clause supports the following time coordinates based on weeks:

- Day of week coordinate composed of a day of week, for example "Tuesday"
- Week of year coordinate composed of a week number, for example "week 15"
- Week day coordinate composed of a week of year coordinate and a day of week, for example "Tuesday week 15"
- Year week coordinate composed of a Gregorian year and a week of year coordinate, for example "2010 week 15."
- Year week day coordinate composed of a Gregorian year, a week of year coordinate, and a day of week, for example "Tuesday 2010 week 15."

The detailed definitions of these time coordinates follow:

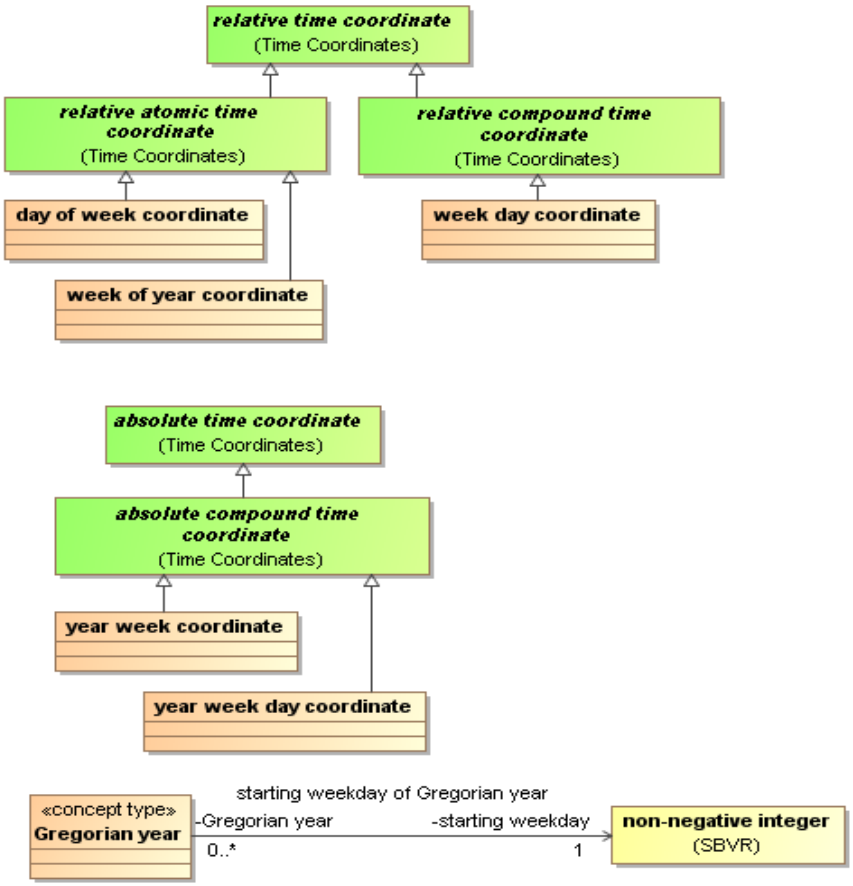

Figure 12.10 - Week Coordinates

day of week coordinate

Definition: relative atomic time coordinate that *indicates* a day of week that *has* the index equal to the index *of* the day of week coordinate

Necessity: Each day of week coordinate *is greater than or equal to* 1.

Necessity: Each day of week coordinate *is less than or equal to* 7.

Example: Wednesday

week of year coordinate

Synonym: calendar week number

Definition: relative atomic time coordinate that *indicates* a week of year that *has* the index equal to the index *of* the week of year coordinate

Definition: number which identifies a calendar week within its calendar year according to the rule that the first calendar week of a calendar year is that one which includes the first Thursday of that calendar year and that the last calendar week of a calendar year is the calendar week immediately preceding the first calendar week of the next calendar year

Dictionary Basis: ISO 8601 (clause 2.2.10)

Necessity: Each week of year coordinate *is greater than or equal to* 1.

Necessity: Each week of year coordinate *is less than or equal to* 53.

Example: week 35

week day coordinate

Definition: relative compound time coordinate that *combines* the set *of* {a week of year coordinate, a day of week coordinate}, and *indicates* a weekday of year that *has* index equal to (7 * (index *of* the week of year coordinate – 1)) + the index *of* the day of week coordinate - 1

Example: Wednesday week 35 *indicates* weekday of year 240

Example: Sunday week 1 *indicates* weekday of year 6

year week coordinate

Definition: absolute compound time coordinate that *combines* the set *of* {a Gregorian year coordinate, a week of year coordinate}, and *indicates* a time period *from* the starting weekday *of* the Gregorian year *indicated by* the Gregorian year coordinate + (7 * (the index *of* the week of year coordinate – 1)) *to* the starting weekday *of* the Gregorian year *indicated by* the Gregorian year coordinate + (7 * (the index *of* the week of year coordinate))

Example: 2010 week 35 *indicates* the time period *from* Gregorian day 634 885 *to* Gregorian day 634 892.

year week coordinate *indicates* time point sequence

Synonymous Form: time point sequence *indicated by* year week coordinate

Definition: the Gregorian year coordinate and the week of year coordinate *of* year week coordinate jointly specify a time point sequence of Gregorian days

Necessity: The Gregorian days scale *is* the time scale *of* time point sequence.

Note: Unlike other time coordinates, this one *indicates* a time point sequence, not a time point. This is because a year week coordinate means a sequence of calendar days rather an individual time point on some time scale. Understanding a year week coordinate as a sequence of calendar days permits comparisons with other time coordinates that can be converted to the Gregorian days scale.

year week day coordinate

Definition: absolute compound time coordinate that *combines* the set *of* {a Gregorian year coordinate, a week of year coordinate, a day of week coordinate }, and *indicates* the Gregorian day that *is* the starting week day *of* the Gregorian year *indicated by* the Gregorian year coordinate, plus (7 * (index *of* the week of year coordinate – 1) + index *of* the day of week coordinate - 1

Example: Wednesday 2010 week 35 indicates Gregorian day 834 647 (starting week day *of* 2010) + 238 (7 * (35 – 1)) + 3 – 1 ➔ Gregorian day 834 887.

Several concepts support the week calendar:

starting week day

Concept Type: role

Definition: the index *of* the Gregorian day of year that *is* 3 days *before* the first "Thursday" *of* some Gregorian year

Definition:	the index *of* the Gregorian day that is the first calendar day of the first week of year of Gregorian year
Note:	This is the index on the Gregorian days scale of the Monday (i.e., day of week 1) of the first week of year of a Gregorian year.

starting week day *of* Gregorian year

Definition:	starting week day *equals* ((ceiling((starting day *of* Gregorian year – starting day *of* 2000) / 7) * 7) + 2 + starting day *of* 2000
Necessity:	The index *of* Gregorian year *is greater than* 1600.
Note:	"Ceiling" in this formula means that the result should be rounded up to the next integer. For example ceiling(4.5) is 5, and ceiling(-4.5) is -4.
Note:	Division ("/") is real division in this formula, not integer division.
Source:	[ISO 8601] says that 1 January 2000 is a Saturday.
Note:	The first week of year of 2000 is the following week of year. The first week day of the first week of year of 2000 is Monday, 2 days after 1 January 2000.
Note:	1 January 2000 is Gregorian day 830 991.
Example:	1 January 2010 is Gregorian day 834 644. The difference between 1 January 2010 and 1 January 2000 is 3 653 calendar days, which rounds up to 522 whole calendar weeks. 521 weeks is equivalent to 3 654 days. Therefore the starting week day of 2010 is 830 991 + 3 654 + 2 days, which is Gregorian day 834 647. (To verify, recognize that Gregorian day 834 647 is 3 days after 1 January 2010, making it 4 January 2010. Any calendar shows that 4 January 2010 is a Monday.

12.3.4 Combining Dates and Times of Day

A date coordinate may be combined with a time of day coordinate to produce a date time coordinate.

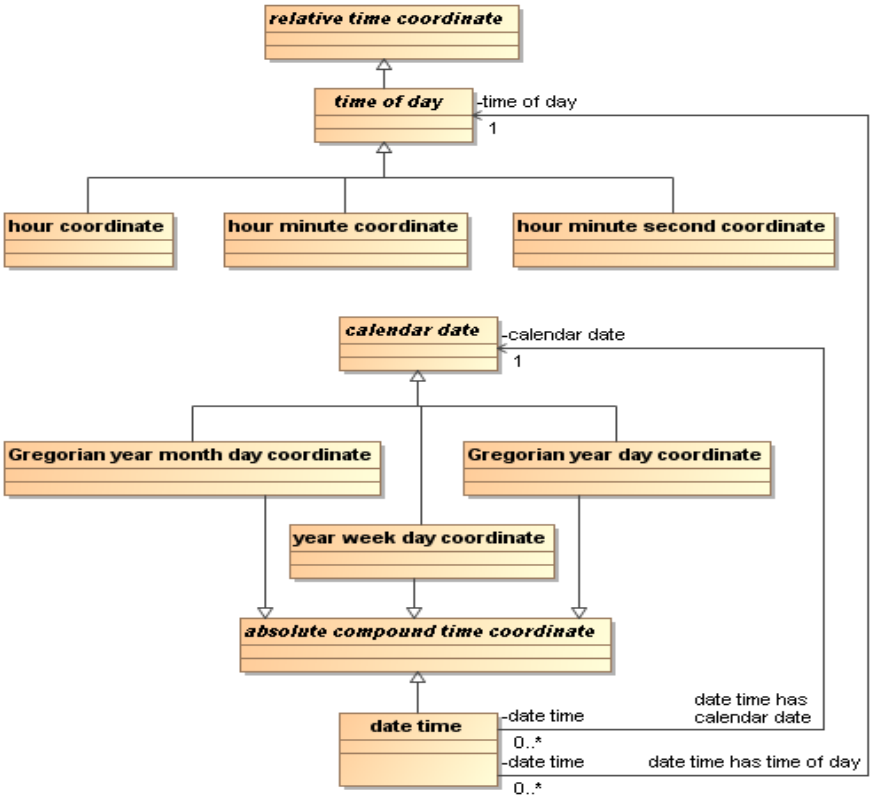

Figure 12.11 - Combination Time Coordinates

calendar date

Synonymous Form:	date coordinate
Definition:	Gregorian year month date coordinate or Gregorian day of year coordinate or year weekday coordinate
Dictionary Basis:	ISO 8601 (2.1.9, 'calendar date')
Note:	Date coordinates may be combined with time offsets, see clause 12.5.
Example:	1989 September 3
Example:	2005 day 49

time of day

Synonymous Form:	time of day coordinate
Definition:	hour coordinate or hour-minute coordinate or hour minute second coordinate
Note:	Time of day coordinates may be combined with time offsets, see clause 12.5.
Example:	3 p.m.
Example:	15:00
Example:	15:00:35

date time

Synonymous Form:	date time coordinate
Definition:	compound time coordinate that *combines* the set *of* {a date coordinate, a time of day time coordinate} and *indicates* the time point that *is on* the combined time scales *of* the time of day time coordinate and the date coordinate, and the time scale *of* the time of day time coordinate *subdivides* the time point that *is indicated by* the date coordinate
Note:	Date time coordinates may be combined with time offsets, see clause 12.5.
Example:	June 9, 1990 5:49:03 p.m.
Example:	13:00 on 1949 day 53
Example:	6 p.m. on 2010 August 6

12.4 Time Scale Comparison and Conversion

Two time points are commensurable (comparable) if and only if they are on the same time scale, or can both be converted to a common time scale. For example, "hour 10" is commensurable with "11:30" because "hour 10" can be converted to a minute of day on the day of minutes scale, and "11:30" is on already that time scale. "hour 10" is not commensurable with "March" because they cannot be converted to any common time scale.

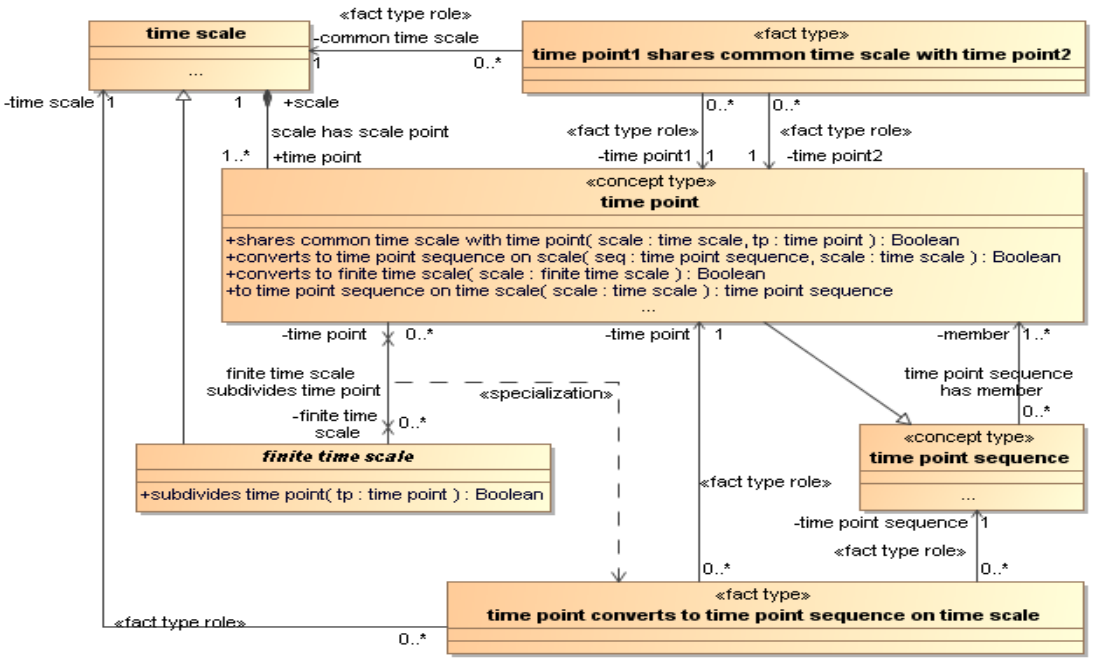

Figure 12.12 - Time Scale Commonality and Conversion

The concept "time point$_1$ *shares* common scale *with* time point$_2$" is used below to declare specific combinations of time points that can be compared if they are converted to particular common time scales. Other combinations are not commensurable.

common time scale

Concept Type: role

General Concept: time scale

time point$_1$ *shares* common time scale *with* time point$_2$

Definition: some time point sequence$_1$ *on* the common time scale *corresponds to* each time period that *instantiates* time point$_1$ and some time point sequence$_2$ *on* the common time scale *corresponds to* each time period that *instantiates* time point$_2$

The concept "time point *converts to* time period *on* time scale" enables conversion of a time point on some time scale$_1$, to a time period on the given time scale. The target time scale always is finer, meaning that it has a granularity that is less than or equal to the granularity of time scale$_1$. This means that time point is equivalent to a time period on time scale$_2$. For example, the Gregorian month that *is indicated by* "January" (on the Gregorian year of months scale) is the time period from Gregorian day of year 1 *through* Gregorian day of year 31 on the Gregorian year of days scale.

The following concept is defined generally below, with each possible conversion explicitly defined in the following sub clauses.

time point *converts to* time period *on* time scale

Definition: time point *converts to* a time point sequence *on* the time scale and the time period *instantiates* the time point sequence

Definition: time point *corresponds to* time period if the time scale$_1$ *of* time point *is* the given time scale

Note: The previous definition enables the case where no conversion is actually needed because the relative time point is already on the target time scale.

Example: The proposition '"9 January" *converts to* time period *on* the Gregorian year of days scale' requires no conversion because "9 January" is already on that time scale.

Necessity: The granularity *of* time scale ≤ the granularity *of* time point.

Note: The specific conversions supported by this document are defined below in verb concepts that have the pattern "time coordinate *converts to* time point sequence *on* time scale." The verb concept defined in this glossary entry uses those verb concepts in a conversion to "time period" rather than "time point set."

Example: The hour of day that *is indicated by* "hour 1" *converts to* the time period that *instantiates* the time point sequence second of day 3600 *through* second of day 7199 *on* the day of seconds scale

The following clauses identify the particular combinations of time point comparisons and conversions that are defined by this specification. Other combinations are invalid. Each clause first specifies which kinds of 'time point' share the same time scale, and then provides definitions for conversion of time points to time periods on another time scale.

12.4.1 Gregorian Indefinite Scale Comparisons and Conversions

These verb concepts enable comparison of time points that are on different indefinite time scales. These are absolute time points, meaning that each *corresponds to* exactly one time interval.

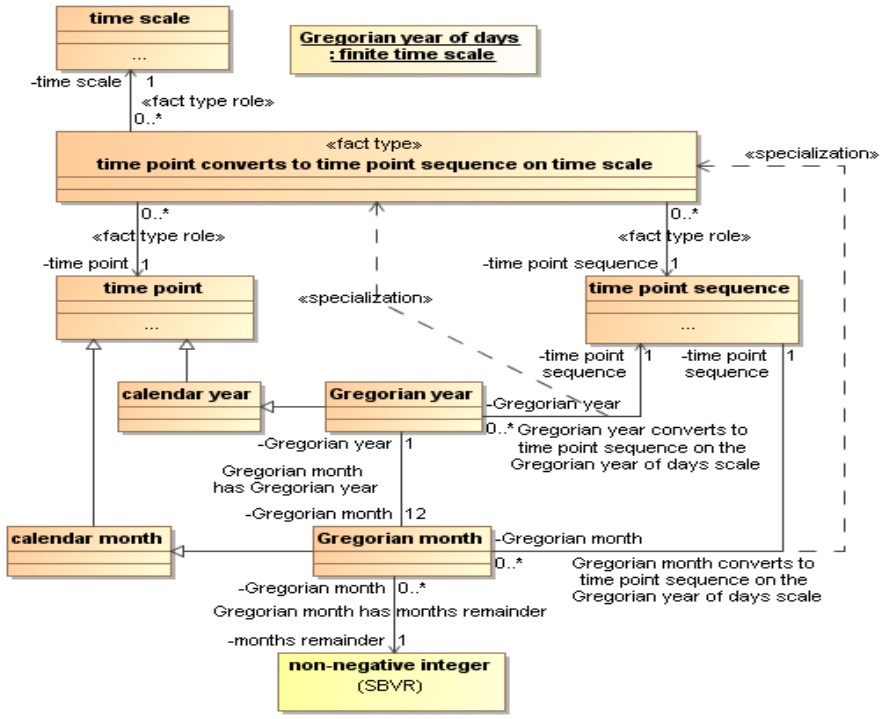

Figure 12.13 - Gregorian Year Conversions

Gregorian year *shares* the Gregorian days scale *with* Gregorian month

Synonymous Form: Gregorian month *shares* the Gregorian days scale *with* Gregorian year

General Concept: time point$_1$ *shares* common time scale *with* time point$_2$

Example: 1979 *shares* the Gregorian days scale *with* June 1990

Gregorian year *shares* the Gregorian days scale *with* Gregorian day

Synonymous Form: Gregorian month *shares* the Gregorian days scale *with* Gregorian year

General Concept: time point$_1$ *shares* common time scale *with* time point$_2$

Example: 1949 *shares* the Gregorian days scale *with* 23 June 1990

Gregorian month *shares* the Gregorian days scale *with* Gregorian day

Synonymous Form: Gregorian month *shares* the Gregorian days scale *with* Gregorian year

General Concept: time point$_1$ *shares* common time scale *with* time point$_2$

Example: June 1990 *shares* the Gregorian days scale *with* 23 June 1990

The following verb concepts support conversion of Gregorian years and Gregorian months to the Gregorian days scale.

Gregorian year *converts to* time point sequence *on* the Gregorian days scale

Definition: the index *of* the first time point *of* the time point sequence *equals* the starting day *of* Gregorian year, and the index *of* the last time point *of* the time point sequence *is* the index *of* the first time point plus 365, plus 1 if the Gregorian year *is* a leap year

Example: The Gregorian year that *is indicated by* "2010" *converts to* Gregorian day 830 991 *through* Gregorian day 831 356 *on* the Gregorian days scale.

Gregorian month *has* Gregorian year

Definition: the index *of* the Gregorian year *equals* 1 plus (Gregorian month divided by 12, using integer arithmetic)

Example: Gregorian month 24108 *has* Gregorian year 2010

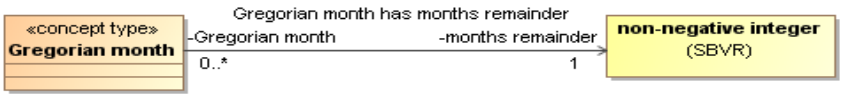

Figure 12.14 - Months Remainder

Gregorian month *has* months remainder

Definition: months remainder equals 1 plus the remainder produced by dividing the index *of* Gregorian month by 12, using integer arithmetic

Gregorian month *converts to* time point sequence *on* the Gregorian days scale

Definition: the Gregorian year *of* Gregorian month *converts to* time point sequence$_1$ *on* the Gregorian days scale, and the months remainder *of* Gregorian month selects a time set from Table 6: Time sets for Gregorian Months, and time point sequence *equals* the first time period of time set plus the index *of* the first time point *of* time point sequence$_1$ – 1 day

Example: The Gregorian month that *is indicated by* "June 2010" *converts to* Gregorian day 831 142 *through* Gregorian day 831 171 *on* the Gregorian days scale.

Table 12.2 - Time sets for Gregorian Months

Gregorian month by month number	Gregorian month by name	time set *on* the Gregorian year of days scale
1	January	{Gregorian day of year 1 *through* Gregorian day of year 31}
2	February	{Gregorian day of year 32 *through* Gregorian day of year 59, Gregorian day of year 32 *through* Gregorian day of year 60}
3	March	{Gregorian day of year 60 *through* Gregorian day of year 90, Gregorian day of year 61 *through* Gregorian day of year 91}
4	April	{Gregorian day of year 91 *through* Gregorian day of year 120, Gregorian day of year 92 *through* Gregorian day of year 121}
5	May	{Gregorian day of year 121 *through* Gregorian day of year 151, Gregorian day of year 122 *through* Gregorian day of year 152}
6	June	{Gregorian day of year 152 *through* Gregorian day of year 181, Gregorian day of year 153 *through* Gregorian day of year 182}
7	July	{Gregorian day of year 182 *through* Gregorian day of year 212, Gregorian day of year 183 *through* Gregorian day of year 213}
8	August	{Gregorian day of year 213 *through* Gregorian day of year 243, Gregorian day of year 214 *through* Gregorian day of year 244}
9	September	{Gregorian day of year 244 *through* Gregorian day of year 273, Gregorian day of year 245 *through* Gregorian day of year 274}
10	October	{Gregorian day of year 274 *through* Gregorian day of year 304, Gregorian day of year 275 *through* Gregorian day of year 305}
11	November	{Gregorian day of year 305 *through* Gregorian day of year 334, Gregorian day of year 306 *through* Gregorian day of year 335}
12	December	{Gregorian day of year 335 *through* Gregorian day of year 365, Gregorian day of year 336 *through* Gregorian day of year 366}

The table shown above is derived from Table 1 of [ISO 8601].

12.4.2 Gregorian Month Comparisons and Conversions

These verb concepts enable comparison of time points that are on the Gregorian year of days scale and the Gregorian year of months scale.

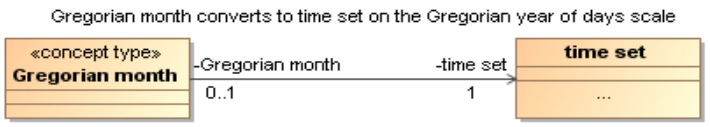

Figure 12.15 - Gregorian Month Conversion

Gregorian month *shares* the Gregorian year of days scale *with* Gregorian day of year

Synonymous Form: Gregorian day of year *shares* the Gregorian year of days scale *with* Gregorian month

General Concept: time point$_1$ *shares* common time scale *with* time point$_2$

Example: "May" can be compared with "day 33" on the Gregorian year of days scale

Because of leap days, Gregorian months *convert to* time sets on the Gregorian year of days scale, rather than to individual time periods. The following verb concepts support these conversions.

Gregorian month *converts to* time set *on* the Gregorian year of days scale

Definition: time set is given in Table 12.2 according to the Gregorian month

Example: The Gregorian month that *is indicated by* "August" *converts to* {Gregorian day of year 213 *through* Gregorian day of year 243, Gregorian day of year 214 *through* Gregorian day of year 244}

12.4.3 Calendar Week Comparisons and Conversions

Comparisons and conversions are also possible among time points that are based on weeks of year, and time points that are based on the Gregorian calendar.

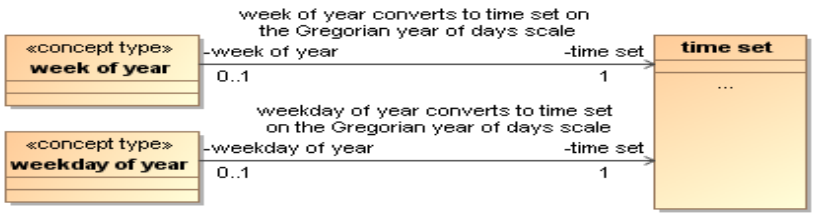

Figure 12.16 - Gregorian Week Conversion

week of year *shares* the Gregorian year of days scale *with* Gregorian day of year

Synonymous Form: Gregorian day of year *shares* the Gregorian year of days scale *with* week of year

General Concept: time point$_1$ *shares* common time scale *with* time point$_2$

Example: "week 33" can be compared with "4 May" on the Gregorian year of days scale

Example: "week 33" can be compared with "day 33" on the Gregorian year of days scale

week of year *shares* the Gregorian year of days scale *with* Gregorian month

Synonymous Form: Gregorian month *shares* the Gregorian year of days scale *with* week of year

General Concept: time point$_1$ *shares* common time scale *with* time point$_2$

Example: "week 33" can be compared with "July" on the Gregorian year of days scale

weekday of year *shares* the Gregorian year of days scale *with* Gregorian day of year

Synonymous Form: Gregorian day of month *shares* the Gregorian year of days scale *with* weekday of year

General Concept: time point$_1$ *shares* common time scale *with* time point$_2$

Example: "Wednesday week 33" can be compared with "4 May" on the Gregorian year of days scale

Example: "Wednesday week 33" can be compared with "day 33" on the Gregorian year of days scale

weekday of year *shares* the Gregorian year of days scale *with* Gregorian month

Synonymous Form: Gregorian month *shares* the Gregorian year of days scale *with* weekday of year

General Concept: time point$_1$ *shares* common time scale *with* time point$_2$

Example: "Wednesday week 33" can be compared with "July" on the Gregorian year of days scale

Each week of year and each weekday of year *converts to* a time set on the Gregorian year of days scale because a week of year can start on any of the first 7 calendar days of a calendar year.

week of year *converts to* time set *on* the Gregorian year of days scale

Definition: time set *equals* (the index *of* the week of year – 1) * 7 + {Gregorian day of year 1, Gregorian day of year 2, Gregorian day of year 3, Gregorian day of year 4, Gregorian day of year 5, Gregorian day of year 6, Gregorian day of year 7}

Example: The week of year that *is indicated by* "week 3" *converts to* {Gregorian day of year 15, Gregorian day of year 16, Gregorian day of year 17, Gregorian day of year 18, Gregorian day of year 19, Gregorian day of year 20, Gregorian day of year 21}.

weekday of year *converts to* time set *on* the Gregorian year of days scale

Definition: time set *equals* the index *of* the weekday of year – 1 + {Gregorian day of year 1, Gregorian day of year 2, Gregorian day of year 3, Gregorian day of year 4, Gregorian day of year 5, Gregorian day of year 6, Gregorian day of year 7}

Example: The weekday of year that *is indicated by* "Friday week 3" *converts to* {Gregorian day of year 26, Gregorian day of year 27, Gregorian day of year 28, Gregorian day of year 29, Gregorian day of year 30, Gregorian day of year 31, Gregorian day of year 32}.

12.4.4 Time of Day Comparisons and Conversions

Hours of day, minutes of day, and seconds of day may be compared with each other.

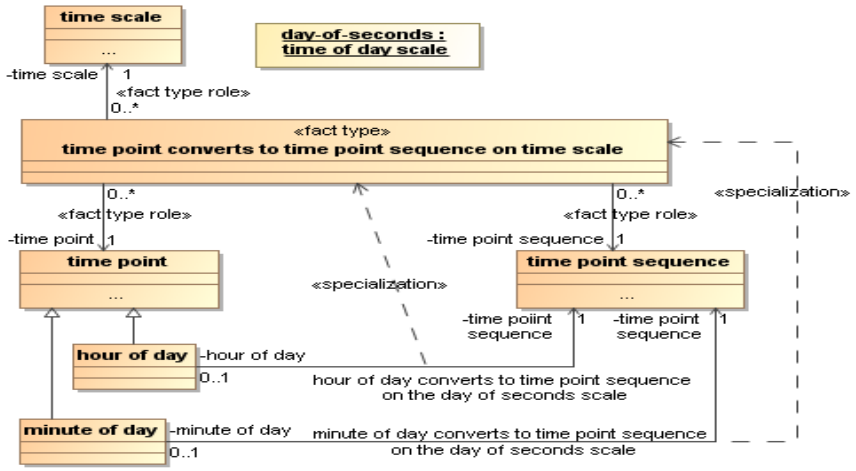

Figure 12.17 - Time of Day Conversions

hour of day *shares* the day of minutes scale *with* minute of day

Synonymous Form: minute of day *shares* the day of minutes scale *with* hour of day

General Concept: time point$_1$ *shares* common time scale *with* time point$_2$

Example: "10 a.m." can be compared with "10:39" on the day of minutes scale

hour of day *shares* the day of seconds scale *with* second of day

Synonymous Form: second of day *shares* the day of seconds scale *with* hour of day

General Concept: time point$_1$ *shares* common time scale *with* time point$_2$

Example: "10 a.m." can be compared with "10:39:42" on the day of seconds scale

minute of day *shares* the day of seconds scale *with* second of day

Synonymous Form: second of day *shares* the day of seconds scale *with* minute of day

General Concept: time point$_1$ *shares* common time scale *with* time point$_2$

Example: "10:39" can be compared with "10:54:48" on the day of seconds scale

Hours of day and minutes of day can *be converted to* the day of seconds scale.

hour of day *converts to* time point sequence *on* the day of seconds scale

Definition: the index *of* the first time point *of* time point sequence *equals* 3 600 times the index *of* the hour of day, and the index *of* the last time point *of* time point sequence *is* the index *of* the first time point plus 3 599

Example: The hour of day that *is indicated by* "hour 0" *converts to* second of day 0 *through* second of day 3 599 *on* the day of seconds scale

minute of day *converts to* time point sequence *on* the day of seconds scale

Definition: the index *of* the first time point *of* time point sequence *equals* 60 times the index *of* minute of day, and the index *of* the last time point *of* time point sequence *is* the index *of* the first time point plus 59

Example: The minute of day that *is indicated by* "1:48" *converts to* second of day 6 480 *through* second of day 6 539 *on* the day of seconds scale.

12.5 Time Zones and Daylight Savings Time

Because of the rotation of the Earth, a separate calendar is needed for each time zone. In order to make local noon (12:00) coincide approximately with the Sun's zenith at the locale, authorities in each locale specify one or more calendars to be used, during different seasons of a year, for commerce in the locale. A locale in which a given calendar is used is called a "time zone." The governing authority over time zones is the national or state government of the locale. Many local calendars are named. For example, Pacific Daylight Time, Eastern Standard Time, British Summer Time. Two or more time zones may have the same name, e.g., Eastern Standard Time in the U.S. (UTC–5 hours) and Australia (UTC+10 hours).

It will not do merely to use UTC with a time offset of the UTC day-of-hours time scale for a local calendar: at any UTC time there is some locale that has a different local date that is one day before or after the UTC date: the date can be different as well as the hour and minute. For example, during periods when daylight savings time is off in Australia (early April to early October), 18:00 UTC (19:00 BST in London) is 04:00 local time the next day in Sydney (UTC+10 hours); 04:00 UTC is 18:00 local time the previous day in Honolulu (UTC–10 hours); Honolulu and Sydney, being 20 hours apart, are on different dates for all but four hours each day (10:00 – 14:00 UTC that day). The approach adopted in this specification is to consider that each time zone has one or two distinguished local calendars.

A local calendar is UTC with a characteristic time offset from UTC by up to ±12 hours. These offsets are usually an integer number of hours or half hours. The nominal offset is zero at the Prime Meridian, +1 hour for each 15º of longitude east of the Prime Meridian, and –1 hour for each 15º of longitude west of the Prime Meridian. '+' means a particular reading of a clock set to the time of the local calendar occurs before a clock that is set to UTC has the same reading; '–' means the local reading occurs after the UTC reading. The duration between corresponding readings is the time offset. The 180º meridian is nominally the International Date Line: a date in locales west of the International Date Line (e.g., longitude 179ºE) is one day ahead of the date in locales east of the International Date Line (e.g., longitude 179ºW).

A complete literal specification of a time interval includes a calendar specification as part of the time coordinate; otherwise there is a 24 hour ambiguity. For example, compare "July 4, 2010 12:00 PDT" to "July 4, 2010 12:00" or "July 4, 2010 PDT" to "July 4, 2010." Note the 24-hour ambiguity when the calendar specification is left out, not knowing where in the world the time is meant.

The intended calendar is often implied by the locale of the utterance of a time coordinate, or by other context, but a calendar specification should be provided explicitly when necessary to remove all doubt. This is especially important in discourses that involve multiple time zones. When time coordinates are used in a discourse without specifying the calendar, it is assumed for purposes of comparison and date-time arithmetic that they are on the same calendar. Time references without calendar specifications in different discourses also without locale references are not prima facie comparable to within less than 24 hours.

A representation of a time offset may be combined with a date coordinate, a time of day coordinate, or a date time coordinate to indicate that the time coordinate is specified according to a calendar for a particular time zone or for daylight savings time. The effect of the time offset is to shift the interpretation of the time coordinate with respect to UTC.

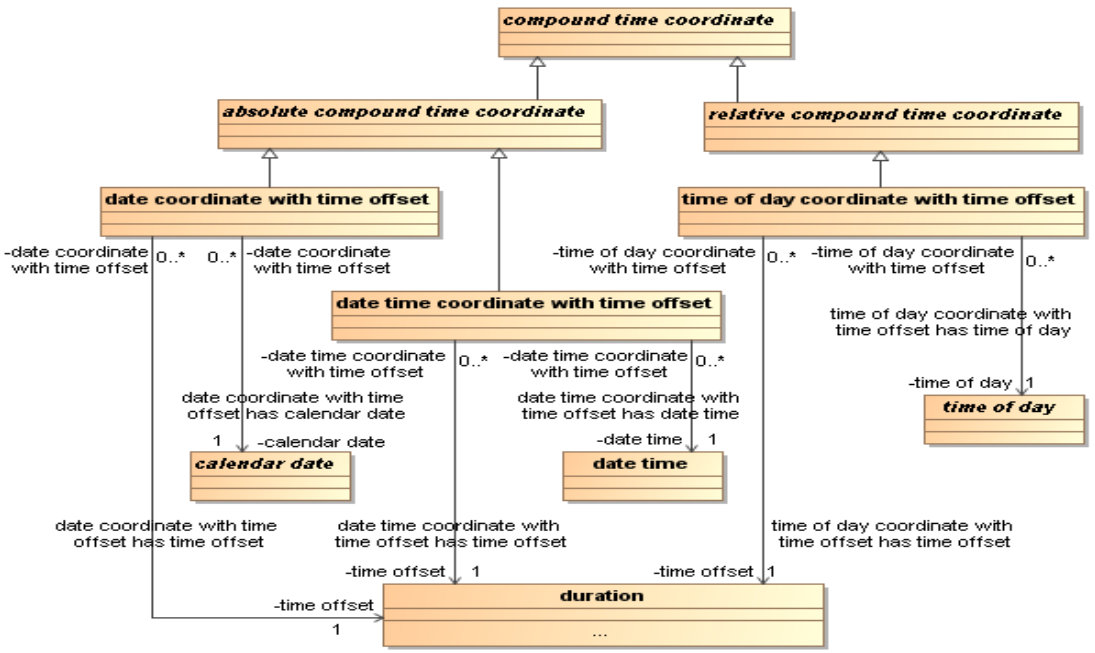

Figure 12.18 - Calendars and Time Offsets

date coordinate with time offset

Definition: time coordinate that combines a date coordinate and a time offset and that *indicates* the time point that *is indicated by* the date coordinate *on* the calendar whose time offset from UTC is the time offset

Note: Time offsets affect the meaning of dates because of the International Date Line.

Example: "July 9 -5" means "July 9" on the calendar specified by time offset "-5".

Example: "July 9 +11" is 22 hours before "July 9 – 11".

time of day coordinate with time offset

Example: time coordinate that combines a time of day coordinate and a time offset and that *indicates* the time point that *is indicated by* the time of day coordinate *on* the calendar whose time offset from UTC is the time offset

Example: "10:00 -5" means "10:00" on the calendar specified by time offset "-5".

Example: "10:00 +11" is 22 hours before "10:00 – 11".

date time coordinate with time offset

Example: time coordinate that combines a date time coordinate and a time offset and that *indicates* the time point that *is indicated by* the date time coordinate *on* the calendar whose time offset from UTC is the time offset

Example: "July 9 10:00 -5" means "July 9 10:00" on the calendar specified by time offset "-5".

Example: "July 9 10:00 +11" is 22 hours before "July 9 10:00 – 11".

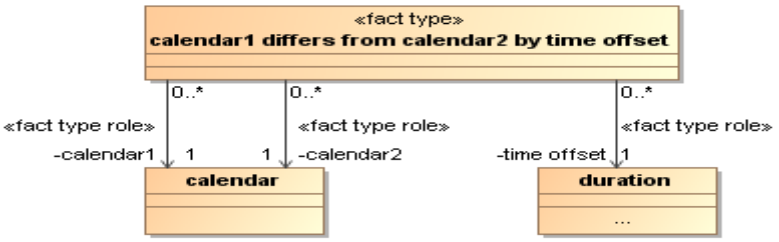

Figure 12.19 - Calendars Differing by a Time Offset

$calendar_1$ *differs from* $calendar_2$ *by* time offset

Definition: time offset equals the $time\ offset_1$ *of* $calendar_1$ minus the $time\ offset_2$ *of* $calendar_2$

Example: Eastern Standard Time (EST) is UTC - 5 hours, and Pacific Standard Time (PST) is UTC - 8 hours. Therefore, EST *differs from* PST *by* 3 hours, and PST *differs from* EST *by* – 3 hours.

Example: India Standard Time (IST) is UTC + 5 hours 30 minutes. Therefore IST *differs from* PST *by* 13 hours 30 minutes.

12.6 Mixed-Base Time Arithmetic

Addition of a duration value to a time coordinate, subtraction of a duration value from a time coordinate, and subtraction of one time coordinate from another all employ "mixed-based time arithmetic." This is an extension of traditional mixed-base arithmetic as employed, for example, in the old-style English currency of pounds, shillings, and pence. The variation of mixed-base arithmetic that is described here accommodates the special issues raised by the nominal time units 'year' and

‘month’, and by the fact that ‘week’ is incommensurate with ‘year’. This procedure is described in text, rather than as a set of SBVR concept definitions, because SBVR is not adapted to defining complex procedures.

Both addition and subtraction apply from the start of a time coordinate. For example,”9 April + 10 hours” is “9 April 10:00”, while “9 April – 10 hours” is “8 April 14:00”.

Mixed-base arithmetic is performed by separately adding or subtracting the individual components of a time coordinate, and then, if necessary, performing a “carry” (for addition) or “borrow” (for subtraction) from the number of the atomic time coordinate that has the next coarser time unit. The result may be either compound or atomic. For example, “9 days 20 minutes = 5 days 13 hours 27 minutes + 2 days 10 hours 53 minutes” by the following steps:

> 27 minutes + 53 minutes ➔ 80 minutes ➔ 20 minutes with 1 hour carry
> 13 hours + 10 hours + 1 hour carry ➔ 24 hours ➔ 0 hours with 1 day carry
> 6 days + 2 days + 1 day carry ➔ 9 days

The following list gives equivalences among most of the precise time units for use in determining when “carries” and “borrows” are needed. Equivalences between years and days, years and weeks, and months and days are discussed below because these are special cases.

> Each minute *is equivalent to* 60 seconds.
> Each hour *is equivalent to* 60 minutes.
> Each day *is equivalent to* 24 hours.
> Each week *is equivalent to* 7 days.
> Each year *is equivalent to* 12 months.

A “carry” is applied if the number of an atomic time coordinate that is formed as an intermediate calculation result is greater than shown in the equivalences given above. To perform a “carry,” divide the number of the intermediate result by the equivalence shown above, add the result to the number of the notional next coarser atomic time coordinate (which may be 0), and set the the number of the finer component to the remainder. Note that “carries” may propagate across multiple components. See the example given above.

A “borrow” is performed if the number of an atomic time coordinate, formed as an intermediate calculation result, is negative. To apply a “borrow,” divide the number of the absolute value of the intermediate result by the equivalence shown above, subtract the result for the number of the notional next coarser atomic time coordinate (which may be 0), and set the the number of the finer component to the remainder. Note that “borrows” may propagate across multiple components. For example, “22 minutes 15 seconds = 35 minutes – 12 minutes 45 seconds” by the following steps:

> 0 seconds – 45 seconds ➔ 15 seconds with 1 minute borrow
> 35 minutes – 12 minutes – 1 minute borrow 22 minutes

The procedure described above works even when an atomic duration value has a number that is greater than an equivalence shown above. For example, “23 hours = 2 days – 25 hours”.

When adding or subtracting values of ‘days’ from time coordinates of the nominal time units ‘year’ and ‘month’, the interpretation of any “carries” or “borrows” depends upon the particular year or month coordinate. For a year coordinate, a “carry” occurs if the number of days exceeds 365 for a common year, and 366 for a leap year, and a “borrow” is made from 365 if the year is a common year, and from 366 otherwise. For example, “2007 day 331 = 2008 – 35 days” but “2006 day 330 = 2007 – 35 days”.

For a month coordinate, “carries” and “borrow” are made according to the following number of days per particular calendar month:

Table 12.3 - Number of Calendar Days per Gregorian Month

Gregorian Month	Equivalent Number of Calendar Days
February	28 in common years, 29 in leap years
April, June, September, November	30
January, March, May, July, August, October, December	31

Note that, in some cases, repeated “carries” or “borrows” may be required across multiple calendar months. For example, “2 March 2010 = 31 January 2010 + 30 days”.

Subtraction is defined for most combinations of two time coordinates. For example, “30 days = 2 March 2010 – 31 January 2010”. However, subtraction of date coordinates that span the end of February is not defined if the calendar years are not specified. For example, “3 days = 2 February – 30 January” but “2 March – 28 February” is either “2 days” or “3 days” depending upon whether these dates are in a leap year or not.

13 Interchange of Duration Values and Time Coordinates (normative)

The Date-Time RFP requests "... a text format for exchanging date and time literals in SBVR interchange files, OWL ontologies, UML models, etc." and goes on to suggest that it should be "based on ISO 8601 as applied in XML Schema." This clause responds to that requirement by specifying how SBVR tools should exchange duration values and time coordinates. This clause assumes that tools will interchange vocabularies using XMI as defined in [SBVR] clause 13.6.

This clause is applicable when one tool that supports this Date-Time Vocabulary transmits a vocabulary to another such tool. This clause is normative for SBVR tools and informative for OWL, CL, UML, or other tools.

[XML Schema] Part 2 defines a datatype named "duration" to represent in XML files what this specification calls duration values. [ISO 8601] defines a lexical representation for durations only as a component of time intervals ([ISO 8601] clause 4.4.3), and [XML Schema] is compatible with it for representing years, months, days, hours, minutes, and seconds. [ISO 8601], but not [XML Schema], supports weeks.

[XML Schema] defines the datatypes listed in Table 13.1 to represent various types of time coordinates. The [XML Schema] lexical representation of these is based on [ISO 8601] but goes beyond [ISO 8601] by defining formats for relative time coordinates such as Gregorian month day coordinate.

Table 13.1 - Relationship between XML Schema and Date-Time time coordinates

XML Schema datatype	corresponding type of time coordinate
dateTime	date time
time	time of day
date	Gregorian year month day coordinate
gYearMonth	Gregorian year month coordinate
gYear	Gregorian year coordinate
gMonthDay	Gregorian month day coordinate
gDay	Gregorian day of month coordinate
gMonth	Gregorian month coordinate

XML Schema does not support several time coordinate types that are defined in [ISO 8601] and in this submission: Gregorian year day coordinate (ISO 8601 clause 4.1.3 "Ordinal date"), year week day coordinate (ISO 8601 8601 clause 4.1.4 "Week date"), and year week coordinate (ISO 8601 8601 clause 4.1.4 "Week date"). Neither XML Schema nor ISO 8601 support these relative time coordinate types: day of week coordinate, week of year coordinate, week day coordinate.

To maximize compatibility with other standards, while supporting the "missing" duration value and time coordinate types listed above, this specification proposes two compliance levels:

1. The first compliance level directly adopts the [XML Schema] "duration" datatype and the date and time datatypes listed in Table 13.1. Tools that implement this compliance level can exploit the features of existing XML parsers and generators. Sub clause 13.1 defines this first compliance level.

2. The second compliance level extends the first compliance level to define interchange formats for the additional duration value and time coordinate types supported by this specification. Sub clause 13.1 defines this second compliance level. Tools that implement this compliance level can use standard XML parsers and generators for the duration, date, and time datatypes defined by [XML Schema] but must implement additional support as described in sub clause 13.1.

13.1 Compliance Level 1 - Interchange Based on XML Schema

Compliance level 1 requires Date-Time tools to support the [XML Schema] “duration” datatype and the datatypes listed in Table 8: Relationship between XML Schema and Date-Time time coordinates. The support must match the definitions of these datatypes in [XML Schema]. The intent is to permit direct use of widely-available XML parsers and generators.

However, the following clauses of [XML Schema] do not apply because this specification describes an approach to ordering of duration values and time coordinates based on duration value sets and time sets:

3.2.6.2 “Order Relation on Duration"“

3.2.7.4 “Order Relation on dateTime”

Implementations should not rely on standard XML software libraries for the order relation on “duration” and “dateTime” values.

13.2 Compliance Level 2 – Additional Interchange Features

Compliance level 2 is a superset of compliance level 1. Compliance level 2 defines additional interchange representations to support the duration values and time coordinate types listed below. These lexical representations are variants of the formats already defined in [ISO 8601]. The design goal is to build upon [ISO 8601] in as simple a manner as possible.

To avoid market fragmentation, tools that implement this compliance level 2 shall support all lexical representations listed in this clause.

The lexical representation for interchanging duration values that include the time unit ‘week’ shall be [PnYnWnDTnHnMnS]. In this representation, the year, day, and time components must conform to the rules defined in [XML Schema] clause 3.2.6.1 for number of digits, value range, use of leading minus sign, reduced precision, and truncation. The number of weeks must be greater than 1. If the number of years, months, days, hours, minutes, or seconds equals zero, the number and corresponding designator may be omitted. Thus, the following examples are all legitimate:

P3W -- three weeks

P3W4D -- three weeks and 4 days

P1Y3W4D-- 1 year and 3 weeks and 4 days

P1Y3W4DT5H-- 1 year and 3 weeks and 4 days and 5 hours

Tools that interchange duration values that may include the ‘week’ time unit should use the “extendedDuration” XML element defined below, to avoid the XML parser changes that would be required if they were interchanged using the standard “duration” datatype. Tools may optionally transmit duration values that do not include weeks using the "duration" datatype defined by [XML Schema]. Since a receiving tool cannot know what duration values may be interchanged, tools should accept all duration value formats in this “extendedDuration” XML element, and should also accept the standard [XML Schema] “duration” datatype.

```
<xs:element name="extendedDuration" type="xs:string">
</xs:element>
```

Table 13.2 lists lexical representations for time coordinate types that are not supported by [XML Schema]. Several of these representations are sourced from [ISO 8601], as shown in the table. These representations should be generated and accepted using the “Extended format” described in [ISO 8601]. In these formats, “YYYY” represents a year number that should have four or more digits; “DDD” is a one- to three-digit day number within the year; “W” is the week designator, “ww” is a one- or two-digit week number within the year; and “D” is a single-digit day number within the week.

Table 13.2 - Interchange Representations for Time Coordinates

time coordinate type	Lexical Representation	Source
Gregorian year day coordinate	YYYY-DDD	[ISO 8601] clause 4.1.3
year week day coordinate	YYYY-Www-D	[ISO 8601] clause 4.1.4
year week coordinate	YYYY-Www	[ISO 8601] clause 4.1.4
Gregorian day of year coordinate	----DDD	described below
day of week coordinate	W-D	described below
week of year coordinate	Www	described below
week day coordinate	Www-D	described below

[ISO 8601] does not describe a representation for the relative time coordinates given in the last four rows of the table, so they are defined here using variations of the standard [ISO 8601] formats. “----DDD” is four leading dashes followed by a one- to three-digit day of the year. This format extends from (and is distinguishable from) the [XML Schema] “gDay” and “gMonth” formats. “W-D” is the week designator with the week number omitted, followed by “-”, followed by a single day of week number. “Www” is the week designator and a week number. “Www-D” is the week designator, followed by a one- or two-digit week number within the year, followed by “-” and a day of week number.

To maintain compatibility with existing XML parsers and generators, tools that generate or receive time coordinates that may have any of the formats listed in this clause should interchange time coordinates using the “extendedDateTime” XML element defined below. Tools may optionally transmit those time coordinates that fit the limitations of [XML Schema] using the standard [XML Schema] datatypes, but may also choose to transmit all time coordinates in the “extendedDateTime” XML element. Since a receiving tool cannot know what time coordinates may be interchanged, receiving tools should accept all time coordinate formats in the “extendedDateTime” XML element and should also accept the standard [XML Schema] datatypes.

```
<xs:element name="extendedDateTime" type="xs:string">
</xs:element>
```

Annex A - Attachments

(normative)

This annex lists the machine-readable attachments that are included in this specification, and identifies which are normative and which are informative. These files are located at: http://www.omg.org/spec/DTV/20120101/.

Table A.1 - Machine-readable Attachments

File	Type	Description	Status
dtv-sbvr.xml	SBVR XMI	SBVR interchange file derived from the text of this specification	normative
dtv-uml.xml	UML	UML model of the SBVR vocabularies	normative, except for the model of Annex D.
dtv.ocl	OCL	OCL constraints stripped out of the text of this specification. The plan is to eventually merge them into the UML model.	normative
dtv.clif	CLIF	CLIF axioms stripped out of the text of this specification. Consistency checked via the Kojeware CLIF Validation Service at http://www.kojeware.com/clif-file-validator. Not yet validated semantically.	normative
sbvr.owl	OWL	OWL model of the parts of SBVR that are needed to support this specification. Validated using Pellet.	informative
sequences.owl	OWL	incomplete OWL model of Annex D. Validated using Pellet.	informative

Annex B - References

(informative)

The authors reviewed a number of standards documents and academic papers. These are listed here.

Allen	Allen, James, *Maintaining Knowledge about Temporal Intervals*, Communications of the ACM, Volume 26, Issue 11, November 1983, pp. 832-843, http://portal.acm.org/citation.cfm?id=358434.
Alspaugh	Alspaugh, Thomas, *Allen's interval algebra*, February 14 2008, http://www.ics.uci.edu/~alspaugh/foundations/allen.html.
BPDM	Object Management Group (OMG), *Business Process Definition MetaModel*, Beta 2, December 2007, http://www.omg.org/cgi-bin/doc?dtc/2007-12-05.
Casati	Casati, Roberto and Varzi, Achille C., *Parts and Places*, MIT Press, 1999.
Davidson	Davidson, Donald, *The Logical Form of Action Sentences*. In: Nicholas Rescher (ed.), *The Logic of Decision and Action*, Pittsburgh: The University Press, pp. 81-95. (1967). Cited in [Kamp, Reyle].
Enhanced Time	Object Management Group (OMG), *Enhanced View of Time Specification*, Version 2.0, January 2008, http://www.omg.org/technology/documents/formal/enhanced_time_service.htm.
Halpin 2007	Halpin, Terry, *Temporal Modeling*, four-part series: • Part 1, February 2007, http://www.brcommunity.com/b332.php. • Part 2, June 2007, http://www.brcommunity.com/b351.php. • Part 3, November 2007, http://www.brcommunity.com/b374.php. • Part 4, April 2008, http://www.brcommunity.com/b411.php.
Halpin 2008	Halpin, Terry, *OWL Time*, 2008, presentation received in private communication.
Haley	Haley, Paul, *Understanding events and processes takes time*, February 19 2008, http://haleyai.com/wordpress/2008/02/19/understanding-events-and-processes-takes-time/.
Hayes	Hayes, Pat, *A Catalog of Temporal Theories*, University of Illinois Technical Report UIUC-BI-AI-96-01, 1995-1996, http://www.ihmc.us/users/phayes/docs/TimeCat96.pdf.
Hobbs 2004	Hobbs, Jerry, *An OWL Ontology of Time*, July 2004, http://www.isi.edu/~hobbs/time/owl-time-july04.txt.
Hobbs 2008	Hobbs, Jerry, *Time Representation*, email on Ontolog-Forum, January 21, 2008, http://ontolog.cim3.net/forum/ontolog-forum/2008-01/msg00336.html.
iCalendar	Internet Engineering Task Force (IETF), *Internet Calendaring and Scheduling Core Object Specification*, RFC 2445, November 1998, http://www.ietf.org/rfc/rfc2445.txt.
IEC 60050-111	International Electrotechnical Committee, *International Electrotechnical Vocabulary - Chapter 111: Physics and chemistry*, Edition 2.0, 1996-07, Available from IEC website.
IKL Guide	Hayes, Pat, Florida Institute for Human and Machine Cognition, *IKL Guide*. Available at www.ihmc.us/users/phayes/ikl/guide/guide.html.

Inter Gravissimas	Pope Gregory XIII, *Inter Gravissimas*, papal bull issued 24 February 1582, prepared in English, Latin, and French by R.T.Crowley for ISO TC154 on 27 December 2002.
International Meridian Conference	International Conference Held at Washington for the Purpose of Fixing a Prime Meridian and a Universal Day, October, 1884. Protocols of the Proceedings available at http://www.gutenberg.org/etext/17759.
International Time	Object Management Group (OMG), *Internationalization, Time Operations, and Related Facilities*, Version 1.0, January 2000, http://www.omg.org:80/technology/documents/formal/internationalization_and_time.htm.
ISO 31-1	International Standards Organization (ISO), *Quantities and units – Part 1:Space and Time*. Replaced by ISO/IEC 80000-3:2007.
ISO 8601	International Standards Organization (ISO), *Data elements and interchange formats – Information interchange – Representation of Dates and Times*, Third edition, December 1, 2004, http://www.iso.org/iso/catalogue_detail?csnumber=40874.
ISO 18026	International Standards Organization (ISO), *Information technology — Spatial Reference Model (SRM) Information technology — Spatial Reference Model (SRM)*, 2009. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=54166
ISO 18629	International Standards Organization (ISO), *Industrial Automation Systems and Integration – Process Specification Language* (PSL), 2004, http://www.iso.org:80/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35431.
ISO 24617-1	International Standards Organization (ISO), *Language resource management – Semantic annotation framework (SemAF) – Part 1: Time and events – Committee Draft*, August 4, 2008, http://www.tc37sc4.org/new_doc/iso_tc37_sc4_n269_ver10_wg2_24617-1_semaf-time_utf8.pdf.
ISO/IEC 80000-3	International Standards Organization (ISO), *Quantities and units -- Part 3: Space and time*, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=31888
Kamp, Reyle	Kamp, Hans and Reyle, Uwe, *From Discourse to Logic*, Kluwer Academic Publishers (1993).
KnowGravity	KnowGravity Inc., *CASSANDRA/xUML User's Guide*, April 2008.
Lee	Lee, Kiyong et. al., *ISO-TimeML and its Applications*, August 24, 2007, http://www.tc37sc4.org/new_doc/iso_tc37_sc4_N385_wg2_iso-timeml_provo2007_beamer_utf8.pdf.
MARTE	Object Management Group (OMG), *UML Profile for Modeling and Analysis of Real-time and Embedded Systems*, Beta 1, August 2007, *http://www.omg.org:80/cgi-bin/doc?ptc/2007-08-04*.
Menzels	Menzels, Chris, *Actualism*, article in the *Stanford Encyclopedia of Philosophy*, December 8, 2008.
Mueller	Mueller, Erik T., *Common Sense Reasoning*, Morgan Kaufmann, 2006, ISBN 978-0-12-369388-4.
NTP	Internet Engineering Task Force, RFC 1305 *Network Time Protocol* (Version 3) http://tools.ietf.org/pdf/rfc1305. (RFC 5905 is a proposed update of RFC 1305.)
OWL Time	World Wide Web Consortium (W3C), *An OWL Ontology of Time*, 27 September 2006, http://www.w3.org/TR/owl-time/.
OWL Time Home	*OWL Time (formerly DAML-Time)*, "home page", http://www.isi.edu/~hobbs/owl-time.html.
Pan	Pan, Feng, *Representing Complex Temporal Phenomena for the Semantic Web and Natural Language*, PhD Thesis, 2007, http://www.isi.edu/~hobbs/time/pub/pan-phdthesis.pdf.

Parsons	Parsons, Terence, *Events in the Semantics of English: a study in subatomic semantics*, MIT Press, 1990, ISBN 0-262-16120-6, http://www.humnet.ucla.edu/humnet/phil/faculty/tparsons/Event%20Semantics/download.htm
QUDV	Object Management Group (OMG), *SysML 1.2 Annex C.5, Quantities, Units, Dimensions, Values*, http://www.omgwiki.org/OMGSysML/doku.php?id=sysml-qudv:quantities_units_dimensions_values_qudv
QUOMOS	OASIS *Quantities and Units of Measure Ontology Standard* (QUOMOS) Technical Committee, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=quomos
SBVR	Object Management Group (OMG), *Semantics of Business Vocabulary and Business Rules* (SBVR), v1.0, January 2008, OMG document formal/08-01-02, http://www.omg.org/spec/SBVR/1.0/.
Schedulability	Object Management Group (OMG), *UML Profile for Schedulability, Performance, and Time Specification*, Version 1, January 2005, http://www.omg.org:80/technology/documents/formal/schedulability.htm.
SI	Bureau International des Poids et Mesures (BIPM), *The International System of Units*, 8th edition, 2006, http://www.bipm.org/utils/common/pdf/si_brochure_8.pdf.
Simons	Simons, Peter, *Parts: A Study in Ontology*, Oxford University Press, 1987.
SysML	Object Management Group (OMG), *Systems Modeling Language V1.0*, September 2007, http://www.omg.org:80/technology/documents/formal/sysml.htm.
TimeML	TimeML Working Group, *Semantic Annotation: A TimeMLCase StudyISO/*, January 8, 2007, http://www.tc37sc4.org/new_doc/ISO_TC37_SC4_N337_WG2_ISO-TimeML_Tilburg2007.pdf.
Time Services	Object Management Group (OMG), *Time Service Specification*, V1.1, May 2002, http://www.omg.org:80/technology/documents/formal/time_service.htm.
UML Time	Object Management Group (OMG), *CommonBehaviors::SimpleTime* package of *UML SuperStructure*, V2.1.2, November 2007, pp , http://www.omg.org/spec/UML/2.1.2/Superstructure/PDF.
VIM	International Standards Organization/International Electrotechnical Commission (ISO/IEC), *International Vocabulary for Metrology – Basic and General Concepts and Associated Terms* (VIM), 3rd edition, JCGM 200:2008 http://www.bipm.org/utils/common/documents/jcgm/JCGM_200_2008.pdf
XML Schema	World Wide Web Consortium (W3C) Recommendation, *XML Schema Part 2: Datatypes Second Edition*, 28 October 2004, http://www.w3.org/TR/xmlschema-2/.

Annex C - EU-Rent Date-Time Example

(informative)

This example is built upon the EU-Rent car rental use case described in Annex E of the SBVR specification. The purpose is to illustrate various aspects of the Date-Time vocabulary, specifically:

1. Vocabulary entries that model dates and durations, and the relationship of these to states of affairs
2. Rules that illustrate:
    a. Comparisons of dates
    b. Comparisons of durations
    c. Date-time arithmetic
    d. Literal dates and times
3. Vocabulary entries that model repeating dates

The goal here is not to redo all aspects of Annex E that mention dates and times, but only to show enough examples to illustrate this date-time vocabulary. Refer to Annex E to understand how the vocabulary entries and rules given below relate to the rest of EU-Rent.

C.1 EU-Rent's Common Vocabulary

Annex E.2.3.3 has a short “Common Vocabulary” used as a basis for the rest of Annex E. The Common Vocabulary includes a number of date and time concepts. The following table shows how the concepts in Annex E.2.3.3 are addressed in this Date-Time Vocabulary:

Table C.1 - Relating EU-Rent Date-Time Concepts

Annex E.2.3.3 Concept	Date-Time Concept
actual date/time	(not standardized)
date	date
date/time	time point
date/time$_1$ *is after* date/time$_2$	time interval$_1$ *is after* time interval $_2$
date/time$_1$ *is before* date/time$_2$	time interval$_1$ *is before* time interval$_2$
date/time *is in the future*	time interval *is in the future*
date/time *is in the past*	time interval *is in the past*
duration	duration
duration$_1$ *is at most* duration$_2$	quantity$_1$ *is at most* quantity$_2$
duration *is measured in* time unit	atomic quantity value *has* measurement unit
end date/time	(not standardized)
period	time interval
period *has* duration	time interval *has* duration
period *has* end date/time	time interval *ends on* time point
period *has* start date/time	time interval *starts on* time point
period$_1$ *overlaps* period$_2$	time interval$_1$ *overlaps* time interval$_2$
start date/time	(not standardized)
state of affairs *occurs after* date/time	situation model *occurs for* time interval$_1$ and time interval$_1$ *follows* time interval$_2$
state of affairs *occurs at* date/time	situation model *occurs for* time interval
state of affairs *occurs before* date/time	situation model *is before* time interval
state of affairs$_1$ *occurs before* state of affairs$_2$ *occurs*	situation model$_1$ *precedes* situation model$_2$

C.2 Describing Dates and Durations

Annex E.2.3.3.1 defines fixed period and variable period concepts as subtypes of period (time period). Here are a few examples, showing how the E.2.3.3.1 concepts can be recast using the new Date-Time Vocabulary.

fixed period

Definition:	time period that cannot *be changed*
Example:	Period in the past, e.g., the OMG Burlingame meeting time.
Example:	Period defined by clock or calendar, e.g., “first ten days in May.”
Example:	Period in the future fixed by fiat, e.g., trip for which you have bought air tickets that cannot be rescheduled or refunded.

variable period

Definition: time period that *can be rescheduled*

Example: period *of* an EU-Rent rental

actual start date/time

Definition: time point that *starts* a variable period that *occurs*

Note: Annex E does not provide a definition for "actual start date/time;" this definition is composed for this example.

scheduled start date/time

Definition: time point that *starts* a variable period

Note: Annex E does not provide a definition for "scheduled start date/time;" this definition is composed for this example.

variable period *has* actual start date/time

Necessity: Each variable period *has* at most one actual start date/time.

Necessity: The actual start time date/time *of* a variable period *is* not *changed*.

variable period *has* scheduled start date/time

Necessity: Each variable period *has* exactly one scheduled start date/time.

Possibility: The scheduled start date/time *of* a variable period *is changed before* the actual start date/time *of* the variable period.

Necessity: The scheduled start date/time *of* a variable period *is* not *changed after* the actual start date/time *of* the variable period.

Note: Additional constraints may be added in specific contexts - e.g., in EU-Rent the cut-off for changing the start date of a points rental is 5 days before its scheduled start date/time.

Here are some more examples, given here both to support some rule examples given below and also to show how the Date-Time Vocabulary can simplify custom vocabularies. In this case, the simplification comes from the use of the time units that are defined in the Date-Time Vocabulary.

rental duration

Concept Type: role

Definition: duration used to calculate a rental charge

rental duration *is measured in* rental time unit

Necessity: Each rental duration *is measured in* a whole number of rental time units.

Necessity: Each rental duration *starts at* the actual start date/time *of* a rental period.

Example: A rental with an end date/time that was 11 days and 7 hours after its start date/time would have a rental duration of 1 x 1-week RTU plus 5 x 1-day RTU. If EU-Rent were to introduce a 3-day RTU, this would change to 1 x 1-week RTU plus 1 x 3- day RTU plus 2 x 1-day RTU.

Note: In the Date-Time Vocabulary, any duration can be expressed as a compound quantity value.

RTU

Definition: hour or day or week or month

Note:	Annex E definition: “time unit that is an atomic (integer) unit of time for which a car can be rented.” Annex E goes on to define several individual concepts (e.g., “Rental Day”) that are not needed, given that this vocabulary has terms such as “nominal day”).
Synonym:	rental time unit
Dictionary Basis:	CRISG [“RTU”]

C.3 Use of Time Points and Durations in Rules

The previous section has several examples (for “variable period *has* actual start date/time” and “variable period *has* scheduled start date/time”) of structural rules associated with dates. Here’s an example of structural rules about durations. This glossary entry is reworked from the description attached to the corresponding entry in Annex E.2.3.3.1 and quoted below.

variable period *has* duration

Description:	Duration of a variable period is measured in one of three ways, depending on what is known at the time of measurement: (1) Before the actual start date/time the duration of a variable period is measured from scheduled start date/time to scheduled end date/time. (2) At any date/time between actual start date/time and actual end date/time the duration of a variable period is measured from actual start date/time to scheduled end date/time. (3) At any date/time after the actual end date/time the duration of a variable period is measured from actual start date/time to actual end date/time (i.e., the period is then fixed).
Necessity:	The duration *is* the duration *of* the time period *from* the scheduled start date/time *of* the variable period *to* the scheduled end date/time *of* the variable period, if today *is before* the actual start date/time *of* the variable period.
Necessity:	The duration *is* the duration *of* the time period *from* the actual start date/time *of* the variable period *to* the scheduled end date/time *of* the variable period, if today *is after* the actual start date/time *of* the variable period and today *is before* the scheduled end date/time *of* the variable period.
Necessity:	The duration *is* the duration *of* the time period *from* the actual start date/time *of* the variable period *to* the actual end date/time *of* the variable period, if today *is after* the actual end date/time *of* the variable period.

Here are some more rules extracted from Annex E. All of these are supported by the Date-Time Vocabulary.

1. It is obligatory that the rental duration *of* each rental *is at most* 90 days.

   This example shows comparison of two durations, one of which is a literal value.
   There is a change from the Annex E original, which read “… *at most* 90 rental days.” This is justified by the definition of “days” as “24 hours,” which matches the original “rental days.”

2. It is necessary that the scheduled pick-up date/time *of* each advance rental *is after* the booking date/time *of* the rental booking that *establishes* the advance rental.

   This rule shows comparison of two dates.

3. It is obligatory that *at* the actual return date/time *of* each in-country rental and each international inward rental the local area *of* the return branch *of* the rental *owns* the rented car *of* the rental.

   A rule that should take effect on the occurrence of an event.

4. It is necessary that the booking date/time *of* a points rental *is at least 5 days before* the scheduled start date/time *of* the rental.

   The "*5 days before*" part of this rule requires date-time arithmetic.

5. It is obligatory that the start date *of* each reserved rental *is in the future*.

   A rule that tests whether a date is in the future.

6. If $rental_1$ *is* not $rental_2$ and the renter *of* $rental_1$ *is* the renter *of* $rental_2$, then it is obligatory that the rental period *of* $rental_1$ does not overlap the rental period *of* $rental_2$.

   This rule tests for the overlap of two time periods.

7. It is necessary that the rental duration *of* each rental that *is the responsibility of* a corporate renter *is* not *greater than* the maximum rental duration *of* the corporate rental agreement that *is available to* the corporate renter.

   Comparison of two durations.

EU-Rent does not seem to have an example of a literal date, so this one is invented for purposes of example. Assume that EU-Rent has changed the cash rental rate, effective 23 March 2009. EU-Rent adopts this rule to apply the new cash rental rate:

> It is obligatory that the rental charge *of* each rental that *starts after* 23 March 2009 *is calculated using* the new cash rental rate.

C.4 Repeating Time Periods

EU-Rent does not use repeating time periods. This example adds to the EU-Rent vocabulary and rules in order to illustrate the Date-Time support for repeating times.

Imagine that EU-Rent offers the ability to schedule repeating rentals, catering to business people who make regularly-scheduled trips and need the same rental arrangement on each trip.

repeating rental
Definition: rental that *has* a time table

repeating rental *has* time table
Definition: Each repeating rental *has* exactly one time table.

next rental
Definition: the first rental that *starts after* this time

last rental
Definition: the latest rental that *starts before* this time

ad hoc repeating rental
Definition: rental whose time table *is* an ad hoc time table

regular repeating rental

Definition: rental whose time table *is* a regular time table

Here are examples of repeating rentals:

1. An irregular repeating rental scheduled for April 1-3 2009, May 12-15 2009, and July 20-23, 2009.
2. A regular repeating schedule scheduled for each Monday in February, 2009.

Annex D - Fundamental Concepts

(informative)

D.1 General

International standards, for example [VIM], [ISO 80000:3], and [ISO 18026] define duration as just one of many quantity kinds, and time scales as one of many kinds of coordinate systems. This permits the formation of derived quantites based on durations (e.g., velocity, which is length / duration), and multi-dimensional coordinate systems that include time as one dimension. Coordinate systems themselves depend upon mathematical concepts, such as sequences and scales. The axioms related to time intervals depend upon mereology concepts.

Unfortunately, there is no existing SBVR vocabulary or ODM ontology that addresses these concepts. The authors recognize that they are out-of-scope for this specification, but felt it necessary to imagine how this Date-Time Vocabulary would fit into a complete schema that addresses them. Annex D summarizes that schema in the form of several SBVR vocabularies.

There are a few existing OMG efforts covering this topic that are referenced in Annex B. The most recent of these is [QUDV], but it models the concept 'quantity' differently than here because of limitations of UML and SysML. In particular, QUDV does not model the distinction between 'quantity' and 'quantity value.'

There is one external group [QUOMOS] that is working in this area, and that is proposed as an OASIS Technical Committee effort called "Quantity and Unit of Measure Ontology Standard (QUOMOS)." As and when [QUOMOS] reaches completion, the contents of this section should be reviewed for possible alignment with [QUOMOS].

This material is informative and is provided to illustrate how the authors think this Date-Time specification fits in a larger context. The team expects that some future effort may work on standardizing these concepts, at which time this specification should be revised to fit properly within the normative version of such concepts.

D.2 Sequences

The 'sequence' concept models ordered collections of things in which the things are ordered by assigning numbers (indices) to them within the collection, as distinct from any particular properties of the things themselves. The model does not preclude the use of properties in creating indices, and it does not require the indices to be consecutive in the general case.

Consecutive sequences add the constraint that successive members have indices that increase by 1. Consecutive sequences provide the mathematical foundation of scales.

There are two somewhat different models of sequence that are in common use. Using UML terminology, we may call them the "composite model" and the "aggregation model." In the composite model, the existence and conceptualization of the members is dependent on the existence and conceptualization of the sequence. In these sequences, the index of a member is intrinsic to the member – its meaning is bound up with its position in the sequence. This is the case with time concepts like months of year or hours of day: 2:00 is the hour of day that occurs immediately after 1:00; its definition depends on the sequence.

In the aggregation model, the members of the sequence have independent existence, with intrinsic properties that are independent of the sequence. The sequence conceptualizes (and imposes) an ordering on the members that is not intrinsic to the members themselves. In these sequences, the indices of the members are extrinsic – the member acquires the index by being included in the sequence, and it can have other indices in other sequences. In some such sequences, a given member can occur more than once. A common example is a list of authorized suppliers in order of preference or total

order volume. Similarly, time intervals exist without clocks, and although they are intrinsically ordered, they only acquire indices when we impose a standard clock and a time offset on them.

The model presented here is general enough to support both models, but each actual sequence will use it differently, depending on the nature of its members. The model below distinguishes between the things that are by definition elements of the sequence – the sequence positions – and things that exist independently and are ordered by the sequence – the members. Time scales, such as clocks and calendars, define the sequence positions of sequences. The application of time scales to the Time Axis, causes the assignment of time intervals as members of the time scales.

Sequences Vocabulary

General Concept:	terminological dictionary
Included Vocabulary:	SBVR Meaning and Representation Vocabulary
Language:	English

D.2.1 General Sequence Concepts

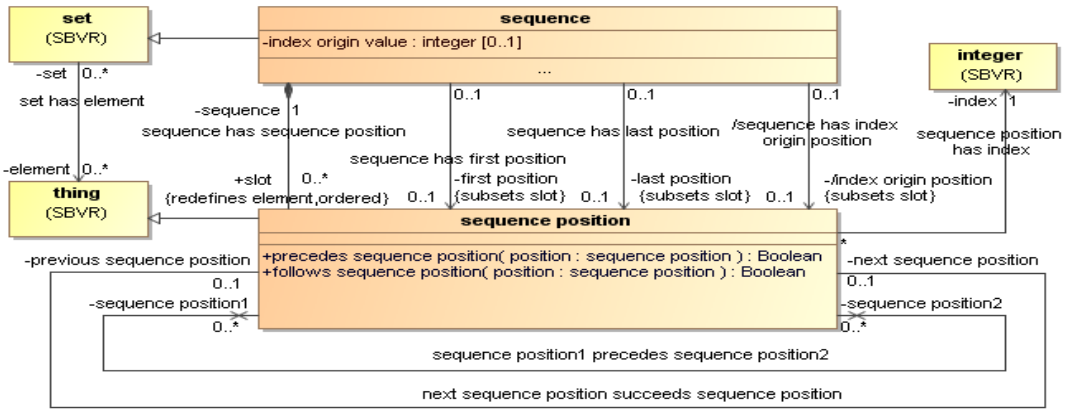

Figure D.1 - Sequences

sequence

Definition:	set whose elements are ordered by their indices
Note:	"Sequence" is a conceptual mechanism for ordering things. A sequence is made up of sequence positions (slots), each of which may have a member. These members are the things that *participate* in the sequence. For convenience, the things that are the members of the sequence positions of a sequence are also called the members of the sequence.
Note:	In the general case, a given thing may *participate* in a sequence more than once, i.e., as more than one member of the same sequence. See 'regular sequence' for a kind of sequence where a thing is constrained to *participate* at most once in the sequence.
Note:	Each sequence defines an ordering on its sequence positions, by assigning an integer index to each sequence position, and using the ordering of the integers to order the members. The index assignment may be based on some natural characteristics of the members, or it may be just sequential position numbers, or it may be some other numbering scheme associated with the meaning of the sequence. In general, the index assignments need

not reflect any natural ordering of the members. That is, the ordering of the members of a sequence can be specific to the sequence concept.

sequence position

Synonym:	slot
Definition:	element of a given sequence
Note:	A sequence is a set of sequence positions. Each sequence position is an element of the sequence that defines it, and no other.
Note:	Each sequence position has an integer index associated with it. The ordering on the sequence is induced on it by the natural ordering of the integers.

sequence *has* sequence position

Synonymous Form:	sequence position *in* sequence
Necessity:	Each sequence position *is of* exactly one sequence.
Possibility:	Some sequence *has* no sequence positions.
Note:	This verb concept is a specialization of SBVR's 'thing *is in* set.'

index

Concept Type:	role
Definition:	integer
Note:	The basis for assigning a particular index to a given sequence position might be a characteristic of the member *of* the sequence position (such as weight, etc.). This technique would order the members by weight, or inversely by weight, depending on the index assignments.
Note:	Negative indices are meaningful for time scales of years that extend before year zero.

sequence position *has* index

Synonymous Form:	index *indexes* sequence position
Definition:	The index is assigned to the sequence position and is used in ordering the sequence positions in the sequence.
Definition:	Each sequence position *has* exactly one index.
Necessity:	If the $index_1$ *of* some $sequence\ position_1$ *of* some sequence *equals* the $index_2$ *of* some $sequence\ position_2$ *of* the sequence then $sequence\ position_1$ *is* $sequence\ position_2$.
CLIF Axiom:	(exists ((s sequence)) (exists (i1 index) (sp1 "sequence position") (i2 index) (sp2 "sequence position")) (if (and ("sequence has sequence position" s sp1) ("sequence has sequence position" s sp2) ("sequence position has index" sp1 i1) ("sequence position has index" sp2 i2) (= i1 i2)) (= sp1 sp2))))
OCL Constraint:	context sequence: inv: self."sequence position"-->forAll(sp1 | self."sequence position"-->forAll(sp2 | indexOf(sp1) = indexOf(sp2) implies sp1 = sp2))

sequence position$_1$ *precedes* sequence position$_2$

Synonymous Form: sequence position$_2$ *follows* sequence position$_1$

Definition: the index *of* sequence position$_1$ *is less than* the index *of* sequence position$_2$.

Note: This is the ordering relation on the sequence positions.

next sequence position *succeeds* sequence position

Synonymous Form: next sequence position *is next after* sequence position

Synonymous Form: sequence position *is just before* next sequence position

Synonymous Form: sequence position *has* next sequence position

Definition: next sequence position *follows* sequence position and the index *of* next sequence position *is less than or equal to* the index *of* each sequence position$_2$ that *follows* sequence position.

CLIF Definition:

```
(forall ((s sequence position) (nsp "sequence position")
        (sp "sequence position"))
 (iff ("next sequence position succeeds sequence position"
       nsp sp)
      (exists ((sp2 "sequence position")
               (ni index) (i2 index))
        (and
         ("sequence position1 precedes sequence position2"
          sp nsp)
         ("sequence position1 precedes sequence position2"
          sp sp2)
         ("sequence position has index" nsp ni)
         ("sequence position has index" sp3 i2)
         (<= ni i2)))))
```

OCL Definition:

```
context "sequence position"
inv: self."sequence position1 precedes sequence position2"
     ( self."next sequence position")
    and self."sequence position2"-->forAll(sp2 |
       self."next sequence position".index <= sp2.index)
```

first position

Concept Type: role

General Concept: sequence position

sequence *has* first position

Definition: the index *of* the first position *is less than or equal to* the index *of* each sequence position *in* the sequence

Necessity: A sequence *has* at most one first position.

Possibility: A sequence may *have* no first position.

Necessity: No sequence position *precedes* the first position *of* each sequence.

last position

Concept Type: role

General Concept: sequence position

sequence *has* last position

Definition:	the index *of* the last position *is greater than or equal to* the index *of* each sequence position *in* the sequence
Necessity:	A sequence *has* at most one last position.
Possibility:	A sequence may *have* no last position.
Necessity:	No sequence position *succeeds* the last position *of* each sequence.

D.2.2 Sequence Members

This sub clause extends the sequence model to accommodate situations in which the sequence position itself is artificial – it represents the role of some thing that exists independently from the sequence.

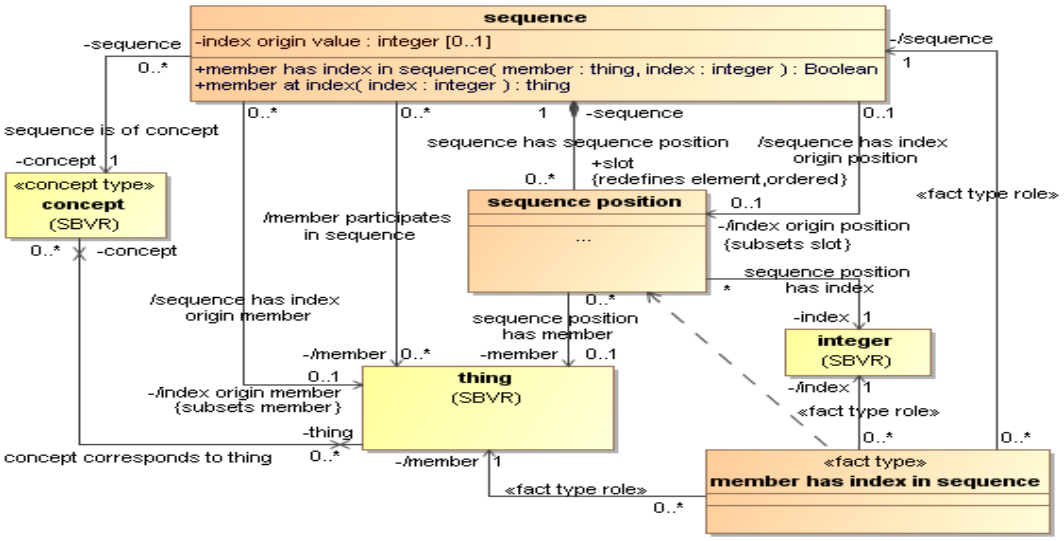

Figure D.2 - Sequence Members

member

Concept Type:	role
General Concept:	thing
Definition:	thing that *is in* a given sequence position, and by extension, any thing that participates in a given sequence

sequence position *has* member

Synonymous Form:	slot *contains* member
Synonymous Form:	member *is in* sequence position
Synonymous Form:	member *in* sequence position
Necessity:	Each sequence position *has* at most one member.
Possibility:	A sequence position *has* no member.
Possibility:	Each thing *is* the member *of* zero or more sequence positions *in* zero or more sequences.

Note: For some sequences, the sequence positions have meaning in their own right, and may or may not have members. For example, the meaning of a scale point is a quantity.

member *participates in* sequence

Synonymous Form: sequence *has* member

Synonymous Form: member *of* sequence

Synonymous Form: member *in* sequence

Definition: the member *is* the member *of* a sequence position *of* the sequence

CLIF Definition:
```
(forall ((s sequence) (member thing))
  (iff ("sequence has member" s member)
     (exists ((sp "sequence position"))
      (and ("sequence has sequence position" s sp)
          ("sequence position has member" sp m)))))
```

OCL Definition:
```
context sequence
def:"member participates in sequence"
    (member: thing, s: sequence)
    : Boolean =
    self."sequence position"-->exists(sp |
     sp.member = member)
```

Note: Things are assigned as members of a sequence to induce a desired ordering relation among the things. Thus, a given set *of* things may be ordered differently in different sequences by their weight, height, arrival time in a queue, service priority, etc.

member *has* index *in* sequence

Synonymous Form: sequence *has* member *with* index

Note: The primary verb concept and the synonymous form given above are “sentential forms” that yield a Boolean result.

Definition: The sequence *has* a sequence position that *has* an index that *equals* the index, and the sequence position *has* a member that *is* the member.

CLIF Definition:
```
(forall ((s sequence) (index integer) (member thing))
 (iff ("member has index in sequence" member index s)
     (exists ((sp "sequence position"))
      ("sequence has sequence position" s sp)
      ("sequence position has index" sp index)
      ("sequence position has member" sp member))))
```

OCL Definition:
```
context sequence
def:"member has index in sequence"
    (member: thing, i: integer, s: sequence)
    : Boolean =
    self."sequence position"-->exists(sp |
      sp.index = index and sp.member = member)
```

Synonymous Form: index *of* member *in* sequence

Note: The Synonymous Form given above is an SBVR “noun form” that yields an index given a member in a sequence.

CLIF Definition:
```
(forall ((s sequence) (index integer) (member thing))
 (iff ("index of member in sequence" member s)
     (exists ((sp "sequence position"))
```

```
("sequence has sequence position" s sp)
(if (and
    ("sequence position has member" sp member)
    ("sequence position has index" sp index))
    index))))
```

OCL Definition:

```
context sequence
def: "index of member in sequence"
    (member: thing, s sequence)
    : integer =
    self."sequence position"-->select(sp |
     sp.member = member).index
```

Synonymous Form: member *with* index *in* sequence

Note: The Synonymous Form given above is an SBVR “noun form” that yields a member given an index and a sequence.

CLIF Definition:

```
(forall ((s sequence) (index integer) (member thing))
 (iff ("member with index in sequence " index s)
    (exists ((sp "sequence position"))
     ("sequence has sequence position" s sp)
     (if (and
         ("sequence position has member" sp member)
         ("sequence position has index" sp index))
         member))))
```

OCL Definition:

```
context sequence
def: "member with index in sequence"
                (index: integer, s sequence)
    : thing =
    self."sequence position"-->select(sp |
     sp.index = index).member
```

sequence *is of* concept

Definition: The concept *corresponds to* each member *of* the sequence.

CLIF Definition:

```
(forall ((member thing) (s sequence))
 (exists ((c concept))
  (iff ("sequence is of concept" s c)
     (if "member participates in sequence" member s)
      ("concept corresponds to instance" c member)))))
```

OCL Definition:

```
context sequence
def: "sequence is of concept"(c: concept)
    : Boolean =
    sequence."sequence position".member-->forAll(m |
            "concept corresponds to instance"(c m))
```

Necessity: Each sequence *is of* at least one concept.

Note: Constraints based on the verb concept ‘ sequence *is of* concept’ limit each member to be an instance *of* the concept. If more than one such constraint is stated for the same sequence, every member must satisfy all such constraints.

Note: Such constraints can be relaxed as needed by specifying that a sequence *is of* any convenient more general concept of the members of the sequence. Since the concept ‘thing’ is a

more general concept of all other object types, a sequence that '*is of* thing' permits members of any type.

D.2.3 Index Origin

index origin member

Concept Type: role

Definition: member that is assigned the index that *is* the index origin value

Note: This is a primitive definition. Either 'index origin member' or 'index origin value' must be defined in terms of the other.

Note: For sequences that have a first member, the first member is usually designated as the index origin member. In a sequence that has no first member, the index origin member is usually determined by association to some real world event or property.

Example: The member with index 1875 (the year of the Convention du Mètre) is the index origin member of the Gregorian years scale.

index origin value

Concept Type: role

General Concept: integer

Note: The index origin value is most commonly either 0 or 1.

Example: The first member of time scales of hours, minutes, and seconds has index origin value 0 because these are counted from 0 by convention.

Example: The first member of time scales of years, months, weeks, and days has index origin value 1 because these are counted from 1 by convention.

index origin position

Concept Type: role

General Concept: sequence position

sequence *has* index origin member

Definition: The index origin member *of* the sequence *is* the member *of* the index origin position *of* the sequence.

Necessity: Each sequence *has* at most one index origin member.

sequence *has* index origin value

Necessity: Each sequence *has* at most one index origin value.

sequence *has* index origin position

Necessity: Each sequence *has* at most one index origin value.

Necessity: The index *of* the index origin position *equals* the index origin value *of* the sequence.

CLIF Axiom:

```
(forall ((s sequence) (p "index order position"))
 (iff ("sequence has index order position" s p)
     (exists ((i1 index) (i2 index))
      (and
       ("sequence position has index" p i1)
       ("sequence has index origin value" s i2)
       (= i1 i2)))))
```

OCL Constraint:
```
context sequence
inv: sequence."index origin value" =
     sequence."index origin position".index
```

D.2.4 Kinds of Sequences

This clause defines various sequence types in order to clarify the distinctions among and meaning of each type.

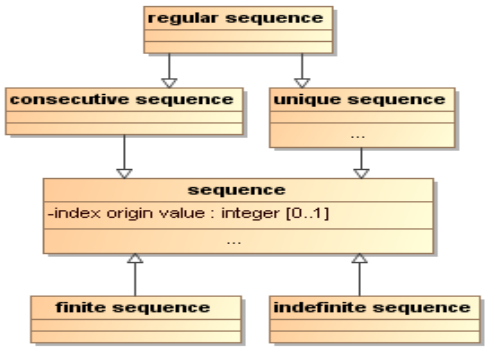

Figure D.3 - Kinds of Sequences

consecutive sequence

Definition: sequence that each sequence position *of* the sequence that *is* not the first position *of* the sequence *is next after* a sequence position$_2$, and the index *of* the sequence position *equals* 1 plus the index *of* the sequence position$_2$

CLIF Definition:
```
(forall ((s sequence))
 (iff ("consecutive sequence" s)
  (forall ((sp1 "sequence position"))
   (if (and
        ("sequence has sequence position" s sp1)
         (not (exists ((first "sequence position"))
               ("sequence has first position" s first)
               (= sp1 first))))
       (exists ((sp2 "sequence position")
          (i1 index) (i2 index))
        (and
         ("sequence is next after sequence position"
          sp1 sp2)
         ("sequence position has index " sp1 i1)
         ("sequence position has index" sp2 i2)))
         (= (+ 1 i2) i1)))))))
```

OCL Definition:
```
context "consecutive sequence"
inv: self."sequence position"-->forAll(sp |
     not (self."first position"-->exists()
       and sp = self."first position")
       implies self."sequence position".indexOf(sp) =
         1 + self."sequence position"
            -->indexOf(sp."previous sequence position"))
```

Note: In a consecutive sequence, the indices of the members are consecutive integers.

unique sequence

Definition: sequence that each member *of* the sequence *is* the member *of* at most one sequence position *of* the sequence

CLIF Definition:

```
(forall ((s sequence))
 (iff ("unique sequence" s)
  (forall ((member thing)
           (sp1 "sequence position")
           (sp2 "sequence position"))
   (if
    (and
     ("sequence has sequence position" s sp1)
     ("sequence has sequence position" s sp2)
     ("sequence position has element" sp1 member)
     ("sequence position has element" sp2 member))
   (= sp1 sp2)))
```

OCL Definition:

```
context sequence
inv: self.member-->forall(m |
       self."sequence position".member-->isUnique (m2 |
         m = m2))
```

regular sequence

Definition: consecutive sequence that *is* a unique sequence

CLIF Definition:

```
(forall ((s sequence)
 (iff ("regular sequence" s)
  (and
    ("consecutive sequence" s)
    ("unique sequence" s))))
```

Note: Regular sequences are the basis of scales (clause D.3).

finite sequence

Definition: sequence that has a cardinality

indefinite sequence

Definition: sequence that *does* not *have* a cardinality

Note: This definition relies on the fact that 'set *has* cardinality' (in MRV) has the Necessity "Each set *has* at most one cardinality." An indefinite sequence has an unknown or unspecified number of elements, hence it does not have a 'cardinality'.

Note: 'Finite sequence' is used in this specification as the basis 'finite time scales', such as the 'Gregorian year of months scale'. 'Indefinite time sequence' is the basis of 'indefinite time scales' such as the 'Gregorian years scale'. The key distinction is that finite time scales have a specified number of sequence positions, whereas the number of sequence positions of indefinite time scales is not known.

Note: Different scientific, religious, and cultural traditions have varying views as to whether there is a first or last calendar year. This specification avoids taking a position about that by using the term 'indefinite sequence' rather than 'infinite sequence'.

D.2.5 Sequence Member Relationships

The following concepts are relationships that a sequence imposes on its members.

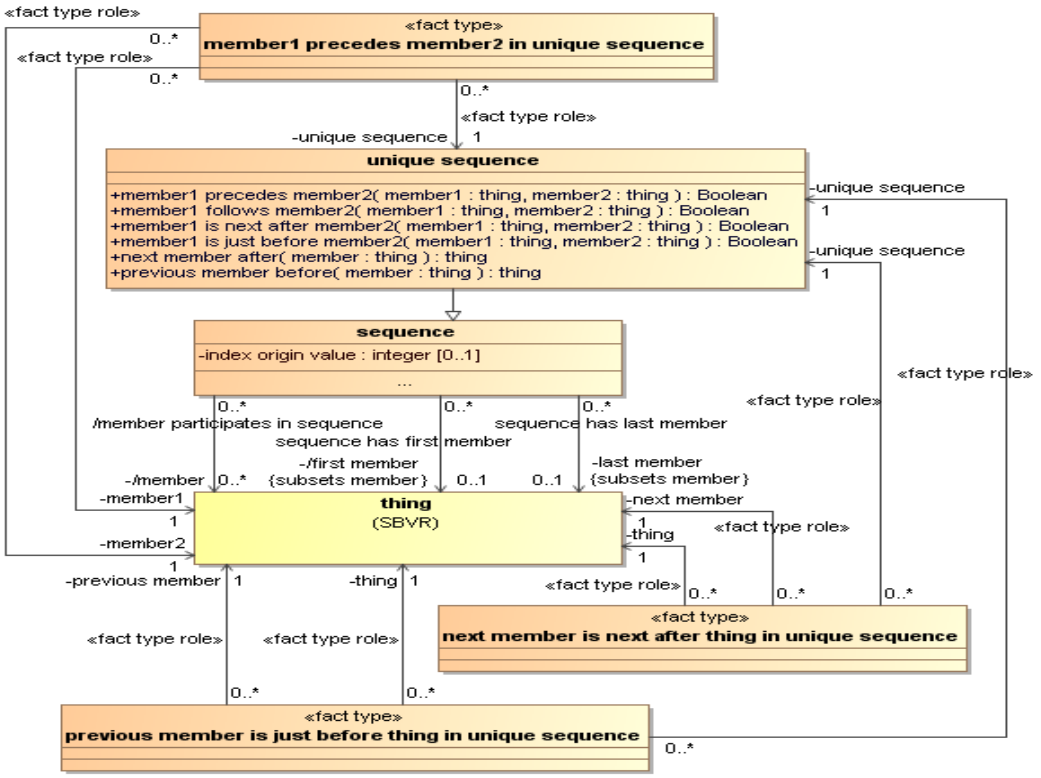

Figure D.4 - Sequence Member Relationships

member$_1$ *precedes* member$_2$ *in* unique sequence

Synonymous Form: member$_2$ *follows* member$_1$ *in* unique sequence

Definition: member$_1$ *participates in* the unique sequence and member$_2$ *participates in* the unique sequence and each sequence position$_1$ *of* member$_1$ *in* the unique sequence *precedes* each sequence position$_2$ *of* member$_2$ *in* the unique sequence

CLIF Definition:

```
(forall ((s sequence)(m1 thing)(m2 thing))
 (iff ("member1 precedes member2 in unique sequence"
       m1 m2 s)
      (and
       ("member participates in sequence" m1 s)
       ("member participates in sequence" m2 s)
       (forall ((sp1 "sequence position")
                (sp2 "sequence position"))
        (if
          (and
           ("sequence position of member" sp1 m1)
           ("sequence position of member" sp2 m2))
```

("sequence position1 precedes
sequence position2" sp1 sp2))))))

OCL Definition:

```
context sequence
def:"member1 precedes member2 in unique sequence"
    (m1: thing, m2: thing) : Boolean =
   self.member-->includes(m1)
    and self.member-->includes(m2)
    and m1."sequence position"-->forall(sp1 |
        m2."sequence position"-->forall(sp2 |
         "sequence position1 precedes
           sequence position2"(sp1, sp2)))
```

first member

General Concept: role

Possibility: thing

sequence *has* first member

Definition: the first member *is the* member *of* the first position *of* the sequence

CLIF Definition:

```
(forall ((s "finite sequence") (m thing)
 (iff ("sequence has first member" s m)
  (exists (first "sequence position"))
   (and
    ("sequence has first position" s first)
    ("sequence position has member" first m)))))
```

OCL Definition:

```
context "sequence"
def: "sequence has first member"
     (s: "sequence") : thing =
    self."first position".member
```

Necessity: A sequence *has* at most one first member.

Note: An indefinite sequence has no first member or no last member.

last member

Concept Type: role

General Concept: thing

sequence *has* last member

Definition: the last member *is the* member *of* the last position *of* the sequence

CLIF Definition:

```
(forall ((s sequence) (m thing)
 (iff ("sequence has last member" s m)
  (exists (last "sequence position"))
   (and
    ("sequence has last position" s last)
    ("sequence position has member" last m))))))
```

OCL Definition:

```
context "sequence"
def: "sequence has last member
     (s: sequence) : thing =
     self."last position".member
```

Necessity: A sequence *has* at most one last member.

Note: An indefinite sequence has no first member or no last member.

next member

Concept Type: role

General Concept: thing

next member *is next after* thing *in* unique sequence

Synonymous Form: thing *has* next member *in* unique sequence

Definition: thing *is* the member *of* exactly one sequence position *in* the unique sequence and next member *is* the member *of* some sequence position *of* the unique sequence that *succeeds* the sequence position *of* the thing

CLIF Definition:

```
(forall ((s sequence) (nm thing) (m thing))
  (iff
   ("next member is next after thing in unique sequence"
    nm m s)
   (exists (sp "sequence position"))
    (and
     ("sequence has sequence position" s sp)
     ("sequence position has member" sp m)
     (forall (sp2 "sequence position"))
       (if
        (and
         ("sequence has sequence position" s sp2)
         ("sequence position has member" sp2 m))
        (= sp2 sp))
     (exists (np "sequence position"))
      (and
       ("sequence has sequence position" s np)
       ("next sequence position succeeds sequence position" np sp)
       ("sequence position has element" np nm)))
    ))
```

OCL Definition:

```
context "unique sequence"
def: "next member is next after member in unique sequence"
     (nm: thing, m: thing) : Boolean =
     self."sequence position".member-->count(m) = 1
     and self."sequence position"-->select(member = m).
       "next sequence position".member = nm
```

Note: This fact type is meaningless if the thing does not appear in the unique sequence.

Necessity: The last member *of* each sequence *has* no next member.

CLIF Axiom:

```
(forall ((s sequence) (last thing))
 (if
  ("sequence has last member" s last)
  (not (exists (m thing))
   ("member is next after thing in sequence"
       m last s))))
```

OCL Constraint:

```
context sequence
inv: self."last member"-->exists() implies
    not self."last member"."next member"-->exists()
```

Necessity: Each member that *participates in* a unique sequence and that *has* no next member *in* the unique sequence *is* the last member *of* the unique sequence.

CLIF Axiom:
```
(forall ((s "unique sequence") (m thing)))
 (if
   (and
    ("element participates in sequence" m s)
    (not (exists (nm thing))
     ("next member is next after thing in sequence"
      nm m s)))
   ("sequence has last member" s m))
```

OCL Constraint:
```
context "unique sequence"
inv: self.member-->forAll(m |
      not m."next member"-->exists()
         implies m = self."last member")
```

previous member

Concept Type: role

General Concept: thing

previous member *is just before* thing *in* unique sequence

Synonymous Form: thing *has* previous member *in* unique sequence

Definition: thing *is* the member *of* exactly one sequence position *in* the unique sequence and previous member *is* the member *of* some sequence position *of* the unique sequence that *is just before* the sequence position *of* the thing

CLIF Definition:
```
(forall ((s sequence) (pm thing) (m thing))
 (iff
  ("previous member is just before thing in unique sequence"
   pm m s)
  (exists (sp "sequence position"))
   (and
    ("sequence has sequence position" s sp)
    ("sequence position has member" sp m)
    (forall (sp2 "sequence position"))
     (if
      (and
       ("sequence has sequence position" s sp2)
       ("sequence position has member" sp2 m))
      (= sp2 sp))
    (exists (pp "sequence position"))
     (and
      ("sequence has sequence position" s pp)
      ("next sequence position succeeds sequence position" sp pp)
      ("sequence position has element" pp pm)))
  ))
```

OCL Definition:
```
context "unique sequence"
def: "previous member is just before member in unique
     sequence"(pm: thing, m: thing) : Boolean =
     self."sequence position".member-->count(m) = 1
```

```
    and self."sequence position"-->select(member = m).
      "previous sequence position".member = pm
```

Note: This fact type is meaningless if the thing does not appear in the unique sequence.

Necessity: The first member *of* each sequence *has* no previous member.

CLIF Axiom:

```
(forall ((s sequence) (first thing))
 (if
   ("sequence has first member" s first)
   (not (exists (m thing))
     ("member is just before thing in sequence"
        m first s))))
```

OCL Constraint:

```
context sequence
inv: self."first member"-->exists() implies
    not self."first member"."previous member"-->exists()
```

Necessity: Each member that *participates in* a unique sequence and that *has* no previous member *in* the unique sequence *is* the first member *of* the unique sequence.

CLIF Axiom:

```
(forall ((s "unique sequence") (m thing)))
 (if
   (and
     ("element participates in sequence" m s)
     (not (exists (pm thing))
       ("previous member is just before thing in sequence"
        pm m s)))
   ("sequence has first member" s m))
```

OCL Constraint:

```
context "unique sequence"
inv: self.member-->forAll(m |
      not m."previous member"-->exists()
        implies m = self."first member")
```

D.3 Quantities Vocabulary

Quantities model many of the concepts in VIM.

Quantities Vocabulary

General Concept: terminological dictionary

Included Vocabulary: Sequences Vocabulary

Included Vocabulary: SBVR Meaning and Representation Vocabulary

Language: English

D.3.1 Quantities

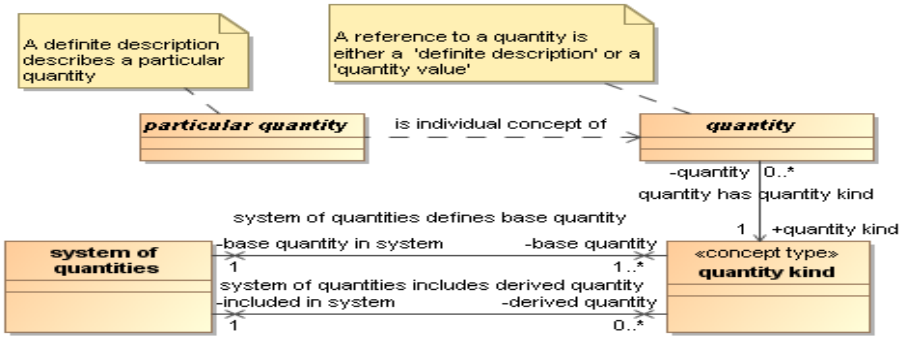

Figure D.5 - Quantities

quantity

Definition:	property of a phenomenon, body, or substance, to which a number can be assigned with respect to a reference
Dictionary Basis:	VIM 1.1 ‘quantity’
Note:	The term ‘quantity’ is used here to refer to the abstraction of the properties – the amount of measurable “stuff” that can be compared between particular quantities. The “height of the Washington Monument” refers to a ‘particular quantity;’ “555 ft 5 inches” refers to a ‘quantity value’.
Note:	This is not the SBVR concept ‘quantity,’ which is deprecated and not used in this model.
Example:	second, kilogram, joule, meter. These are quantities in a general sense, which is what is meant here by ‘quantity.’
Note:	A quantity as defined here is said to be a “scalar” as distinct from a “vector.” However, a vector or a tensor whose components are quantities is also considered to be a quantity.

particular quantity

Definition:	a property that is of an individual thing and is quantifiable as an instance of some quantity kind
Note:	The weight of a given person, the mass of the Earth, the speed of light, and the distance between New York and Paris are said to be “particular quantities.”
Note:	A particular quantity is given by a definite description, which identifies the individual thing and the property. Particular quantities are properties of particular things and are generally expressed by a term for the property and a quantity value.
Note:	Particular quantities appear in fact models as individual concepts that refer to instances of ‘quantity.’ Thus, a conceptual schema might have the fact type “meeting *lasts* duration,” where “duration” is a specialization of “quantity.” (See the note about “duration” under “quantity kind.”) A fact model might include the fact “last Monday’s meeting lasted 2hr 20min.” The definite description “the duration of last Monday’s meeting” defines a particular quantity, an individual concept whose one instance is the quantity (thing) that is quantified by the quantity value “2 hr 20 min.” “2hr 20min” is a compound quantity value of quantity kind “duration.”
Reference Scheme:	A definite description *of* the particular quantity.

quantity kind

Definition: categorization type for 'quantity' that characterizes quantities as being mutually comparable

Concept Type: concept type

Dictionary Basis: VIM 1.2 'kind of quantity'

Example: duration, mass, energy, length

Note: Every instance of 'quantity kind' is also a specialization of 'quantity'. So the concept 'duration' is an instance of 'quantity kind' and it is a specialization of 'quantity', i.e., it is a classifier of actual quantities. But *a given* duration (i.e., the duration of something) is an instance of 'duration' and thus a 'particular quantity,' not an instance of 'quantity kind'. For example, a 'year' is not an instance of quantity kind; it is an instance of quantity, but not a category of quantity.

Note: The quantities "year" and "second" are instances of quantity, and they are both instances of the quantity kind 'duration' and are mutually comparable. Quantities of time given in years and seconds are comparable, although some transformation of quantity values (see below) is needed to compare them. Similarly 'metre' is an instance of 'length,' and 'foot' is another instance of 'length' that is comparable to 'metre,' although conversions are required when comparing values. But 'metre' is not comparable to 'second', because 'length' and 'duration' are disjoint quantity kinds. Only quantities of the same kind are mutually comparable.

Note: All duration quantities are comparable regardless of the role they play – the particular properties they instantiate. The duration of the warranty on an automobile can be compared with the expected life of the battery, even though those are very different particular quantities. Similarly, the height of a tower can be compared to the distance one can see from the top, because they are both length quantities, even though they are unrelated properties.

Note: The concept 'height' is a role of quantities of the quantity kind 'length'. In principle, 'height' could be considered a category of 'quantity' (a sub-category of 'length') and therefore its own 'quantity kind.' The concept 'range of a weapon' is a different role of length quantities. If we want to treat the height of a target as comparable to the range of a weapon, it is inadvisable to treat height and range as different quantity kinds. This idea is the basis for the 'system of quantities' concept.

quantity *has* quantity kind

Definition: quantity is an instance of the category of quantity that is the quantity kind

Necessity: Each quantity *has* exactly one quantity kind.

CLIF Axiom:
```
(forall ((quantity quantity))
 (= (count ("quantity has quantity kind" quantity))
    1))
```

Example: hour (the duration) is an instance of 'duration' – a specific quantity of time. So the quantity kind of 'hour' is 'duration'.

system of quantities

Definition: set of quantities together with a set of non-contradictory equations relating those quantities

Dictionary Basis: VIM 1.3 'system of quantities'

system of quantities ***defines*** **base quantity**

base quantity

Definition: quantity kind in a conventionally chosen subset of a given system of quantities, where no subset quantity can be expressed in terms of the others

Concept Type: role

Dictionary Basis: VIM 1.4 'base quantity'

Example: The International System of Quantities (ISQ) comprises these base quantities (with their SI base measurement units):

length (meter)
mass (kilogram)
duration (second)
electric current (ampere)
thermodynamic temperature (kelvin)
amount of substance (mole)
luminous intensity (candela)

These base quantities are not mutually comparable. All quantities of any one of these kinds are, however, mutually comparable. See also "quantity kind."

system of quantities ***includes*** **derived quantity**

derived quantity

Definition: quantity kind, in a system of quantities, that is not a base quantity of the system but may be defined in terms of base quantities of the system

Dictionary Basis: VIM 1.5 'derived quantity'

Example: velocity (`length/time`), mass density (`mass/length`^{3})

D.3.2 Measurement Units

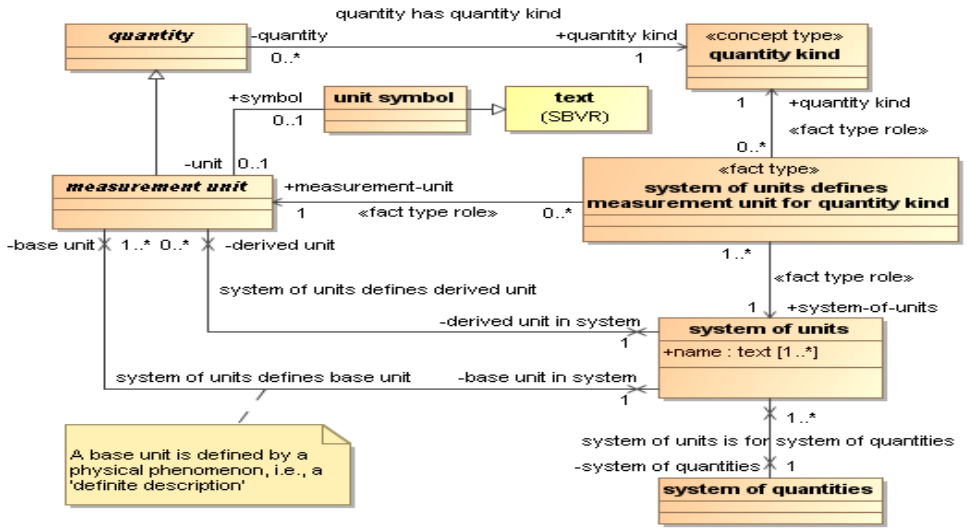

Figure D.6 - Measurement Units

measurement unit

Definition: quantity, defined and adopted by convention, with which any other quantity of the same kind can be compared to express the ratio of the two quantities as a number

Dictionary Basis: VIM 1.9 ‘measurement unit’

Example: week, day, hour, minute, second, kilogram, joule, meter

system of units

Definition: a set of measurement units associated with a system of quantities, together with a set of rules that assign one measurement unit to be the base unit for each base quantity in the system of quantities and a set of rules for the derivation of other units from the base units.

Example: The International System of Units (SI) is a system of units.

system of units *is for* system of quantities

Necessity: Each system of units *is for* exactly one system of quantities.

CLIF Axiom:

```
(forall (("system of units" "system of units"))
      (= (count
         ("system of units is for system of quantities"
           "system of units"))
        1))
```

system of units *defines* measurement unit *for* quantity kind

Definition: The system of units identifies the measurement unit as the reference measurement unit for the quantity kind.

Note: A system of units defines one base unit for each base quantity in the system of quantities that it is for. It may define additional measurement units (derived units) for the same

quantity kinds. It may define derived units for derived quantities, or it may define a mechanism for expressing derived units.

system of units *defines* base unit

base unit

Concept Type:	measurement unit that *is defined by* a system of units to be the reference measurement unit for a base quantity
Concept Type:	role
Dictionary Basis:	VIM 1.10 ‘base unit’
Note:	Quantity units that are not base units are derived units.
Example:	See the example SI units under “base quantity”.

system of units *defines* derived unit

derived unit

Definition:	measurement unit for a derived quantity
Dictionary Basis:	VIM 1.11 ‘derived unit’
Note:	Every derived unit is defined in terms of base units
Example:	1 minute = 60 seconds
Example:	1 stere = 1 metre^3
Example:	1 inch = 0.0254 metre

D.3.3 Quantity values

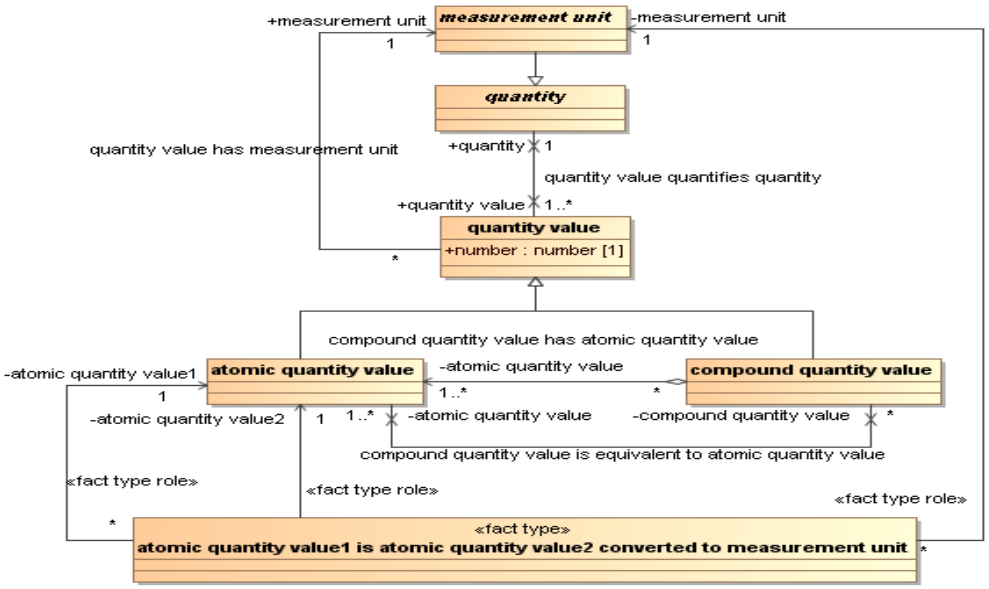

Figure D.7 - Class Diagram for Quantity Values

quantity value

Definition: atomic quantity value or compound quantity value

CLIF Definition:
```
(forall ((thing thing))
 (iff ("quantity value"(thing))
    (or ("atomic quantity value"(thing))
       ("compound quantity value"(thing)))))
```

Note: No OCL definition is provided here because the relationships among quantity value, atomic quantity value, and compound quantity value are shown in the UML diagram.

quantity value *quantifies* quantity

Synonym: quantity *is quantified as* quantity value

Synonym: quantity value *of* quantity

Synonym: quantity value *expresses* quantity

Definition: The quantity value gives the magnitude of the quantity.

Note: For an atomic quantity value, the ratio of the quantity to the measurement unit *of* the quantity value *is* the number *of* the quantity value

Possibility: More than one quantity value may *quantify* a particular quantity.

Example: The duration of a meeting is a particular quantity that might *be quantified as* "1 hour" or as "6 minutes".

atomic quantity value

Definition: number and measurement unit together giving magnitude of a quantity

Dictionary Basis: VIM 1.19 'quantity value'

Note: The quantity expressed by an atomic quantity value is the quantity whose ratio to the measurement unit is the number.

Example: 2 days, 3.5 hours, 150 lb, 45.5 miles. See also the note under "measurement unit".

atomic quantity value *has* number

Definition: The number is the ratio of the quantity expressed by the quantity value to the measurement unit *of* the quantity value

Note: In the general case, number is a mathematical real or complex number. For example, the circumference of a circle one metre in diameter is pi metres. Because the number is a ratio, rational fractions are commonly used in stating quantity values. Thus, it is meaningful to say a task took 2.5 days to complete.

Note: When the number is a non-negative integer, it may be thought of as a count of the units in the quantity. But that view only applies to certain measurement techniques, such as the addition of unit weights to a balance, or the count of ticks of a clock.

atomic quantity value *has* measurement unit

Definition: The measurement unit is the reference quantity to which the ratio of the quantity expressed by the quantity value is taken

Example: The measurement unit of "12 days" is 'day'.

atomic quantity value$_1$ *is* atomic quantity value$_2$ *converted to* measurement unit

Definition: the quantity that *is quantified as* atomic quantity value$_1$ *is* the quantity that *is quantified by* atomic quantity value$_2$ and the measurement unit *of* atomic quantity value$_1$ *is* measurement unit

CLIF Definition:
```
(forall (("atomic quantity value 1" "atomic quantity value")
         ("atomic quantity value 2" "atomic quantity value")
         ("measurement unit" "measurement unit"))
 (iff (converted "atomic quantity value 1"
                 "atomic quantity value 2"
                 "measurement unit")
      (exists ((quantity quantity)
               ("measurement unit 2" "measurement unit"))
        (and ("quantity value has measurement unit"
              "atomic quantity value 2"
              "measurement unit 2")
             (= "measurement unit" "measurement unit 2")
             (= "atomic value 1" quantity)
             (= "atomic value 2" quantity)))))
```

OCL Definition:
```
context "atomic quantity value"
def: "atomic quantity value1 equals atomic value2 converted
     to measurement unit"(
    "atomic quantity value 1": "atomic quantity value",
    "atomic quantity value 2": "atomic quantity value",
    "measurement unit": "measurement unit") =
    "atomic quantity value 1".quantity =
     "atomic quantity value2".quantity
     and "measurement unit"
        = "atomic quantity value2"."measurement unit"
```

Necessity: For each atomic quantity value$_2$ and each measurement unit, exactly one atomic quantity value$_1$ *is* atomic value$_2$ *converted to* the measurement unit.

CLIF Axiom:
```
(forall (("atomic quantity value 2" "atomic quantity value")
         ("measurement unit" "measurement unit"))
        (= (count ("atomic quantity value1 is atomic
                    quantity value2 converted to
                    measurement unit"
                    "atomic quantity value 2"
                    "measurement unit"))
           1))
```

compound quantity value

Definition: quantity value that is an explicit sum of two or more atomic quantity values expressing quantities of the same kind in different measurement units

Example: 2 hours and (plus) 20 minutes expresses the quantity that may also be expressed as 140 minutes

Note: To convert a compound quantity value to an atomic quantity value (so that it expresses the quantity as a ratio to a single measurement unit), the summands of a compound quantity value must be converted to a common measurement unit. This is only possible when precise measurement units are used in the compound quantity value. Thus, 2 hr 20 min = 140 min,

but ‘a year and a day’ is not precisely expressible in days unless a particular year is given, because of leap years.

compound quantity value *has* atomic quantity value

Definition: The atomic quantity value is one of the summands of the compound quantity value.

Necessity: Each compound quantity value *has* at least two atomic quantity values.

CLIF Axiom:

```
(forall (("compound quantity value"
          "compound quantity value"))
 (>= (count ("compound quantity value has
              atomic quantity value"
             "compound quantity value"))
  2))
```

Note: The CLIF axiom above assumes a ">" predicate that compares two numbers to determine if the first is greater than or equal to the second; a "compound quantity value has atomic quantity value" function that takes a compound quantity value and returns a set of atomic quantity values; and a "count" function that returns the number of elements in a set.

Note: The Necessity given above is represented by cardinality in the UML model.

Example: 5 hours and 30 minutes

compound quantity value *is equivalent to* atomic quantity value

Definition: The sum of the atomic quantity values that are the atomic values of the compound quantity value *converted to* the measurement unit of atomic quantity value is atomic quantity value

Necessity: There exists a conversion factor that converts the measurement unit *of* the atomic quantity value to the measurement unit *of* at least one of the atomic quantity values *of* the compound quantity value.

Necessity: The quantity kind *of* the quantity that *quantifies* each atomic quantity value$_1$ *of* the compound quantity value *is* the quantity kind *of* the quantity *of* the atomic quantity value.

CLIF Axiom:

```
(forall (("compound quantity value"
           "compound quantity value")
          ("atomic quantity value" "atomic quantity value"))
 (iff ("is equivalent to" "compound quantity value"
                          "atomic quantity value")
      (forall ("atomic quantity value 1"
               "atomic quantity value")
       (exists (("quantity 1" quantity)
                ("quantity 2" quantity)
                ("quantity kind 1" "quantity kind")
                ("quantity kind 2" "quantity kind")))
        (and ("compound quantity value has atomic
               quantity value"
              "compound quantity value"
              "atomic quantity value 1")
             ("quantity value quantifies quantity"
              "atomic quantity value 1" "quantity 1")
             ("quantity has quantity kind"
              "quantity 1" "quantity kind 1")
```

("quantity value quantifies quantity"
"atomic quantity value" "quantity 2")
("quantity has quantity kind"
"quantity 2" "quantity kind 2")
(= "quantity kind 1" "quantity kind 2"))))))

OCL Constraint:

```
context "compound quantity value"
inv: self."atomic quantity value"-->forAll(
    "quantity."quantity kind" =
      self."compound quantity value is equivalent to
          atomic quantity value"
       ."quantity"."quantity kind"
```

Example: The compound quantity value "5 days 3 hours 20 minutes" combines the atomic quantity value "5 days" with the (compound) quantity value "3 hours 20 minutes," which itself combines the (atomic) quantity values "3 hours" and "20 minutes". The compound quantity value "5 days 3 hours 20 minutes" *is equivalent to* "7400 minutes".

D.4 Scales

This sub clause applies the concept of "sequences" to "quantities" and "quantity values" to model the everyday concept of "scale". This supports ideas such as "5 days after the full moon" and "the third day of every month" via standard arithmetic and quantification operations applied to the indices of sequences.

Scale Vocabulary

General Concept: terminological dictionary

Language: English

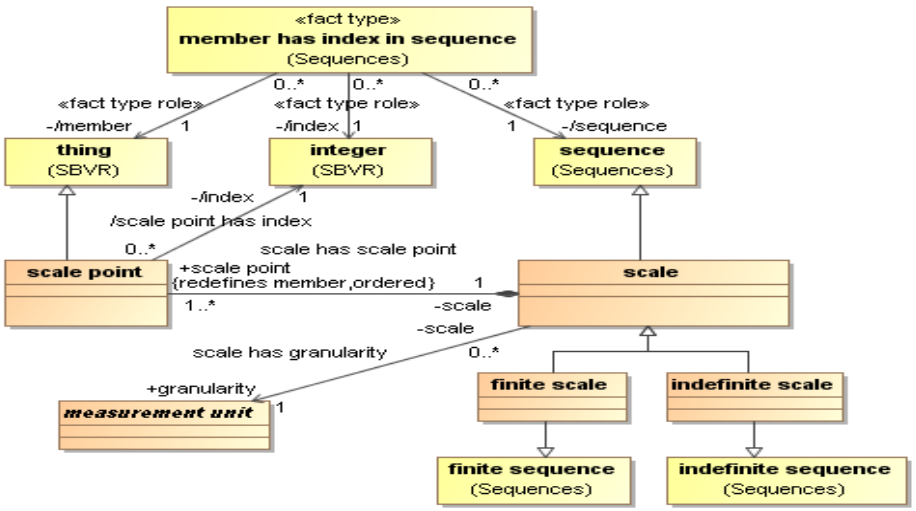

Figure D.8 - Scales

The "black diamond" and {redefines...} on the "scale has scale point" association are wrong.

scale point

Definition:

Concept Type: role

Definition: thing that *is* a member *of* a scale

Note: 'Scale point' is an abstract concept, defined here to serve as a foundational aspect of scales.

Possibility: A scale point *is* a member *of* more than one scale.

Example: 'Tuesday' defined as 'calendar day that *is* the member *of* the week of days scale that *has* sequence position with index 2'. The week of days scale may be applied to 52 or 53 calendar weeks in each calendar year. Therefore there are 52 or 53 Tuesdays in each calendar year.

scale

Synonym: quantity scale

Synonym: quantity-value scale

Definition: regular sequence *of* scale points

Dictionary Basis: VIM 1.27 'quantity-value scale' ordered set of values of quantities of a given kind used in ranking, according to magnitude, quantities of the same kind

Note: The additional semantic of "scale", compared to any other regular sequence, is that each scale has a granularity.

Note: Because a scale is a regular sequence, the indices are consecutive integers. The unit increment of a scale is 1 unit of a given measurement unit, e.g., 1 hour. The scale does not mark fractions of the base measurement unit. Thus, scales are discrete.

Note: A scale can be finite, half-open (having a first or last scale point and indefinitely many lesser or greater ones), or open (extending indefinitely in both the negative and positive directions). For example, the day of hours scale is a finite scale containing 24 hours of day (scale points) running from index 0 to index 23. The Gregorian years scale is an indefinite scale in which the final Gregorian year (scale point) does not exist or at least is not known.

Note: Fractional magnitudes can be interpolated on an analog scale, but not on a discrete scale. The least-significant-digit of number used in a quantity value gives the resolution of the scale used to interpret the expression, i.e. the granularity measurement unit for that scale. For example, 2.345 seconds would be interpreted as 2345 milliseconds (using a scale whose granularity is millisecond).

Reference Scheme: The measurement unit that *is* the granularity *of* the scale.

scale *has* scale point

finite scale

Definition: Scale that *is* a finite sequence.

indefinite scale

Definition: Scale that *is* an indefinite sequence.

granularity

Synonym: resolution

Concept Type: role

Definition: measurement unit that is associated with a scale as the unit of increment of the scale

scale *has* granularity

Necessity: Each scale *has* exactly one granularity.

D.5 Mereology

Mereology is the study [Simons] [Casati] of the relationships among whole things and their parts. This specification relies upon the following mereology axioms, among others, to define the properties of time intervals.

Mereology Vocabulary

General Concept: terminological dictionary

Language: English

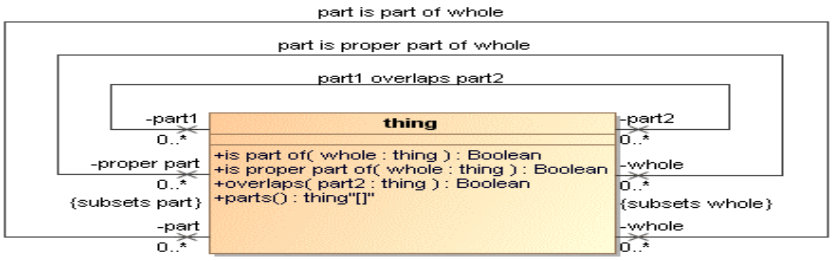

Figure D.9 - Mereology

whole

Concept Type: role

General Concept: thing

part

Concept Type: role

General Concept: thing

part *is part of* whole

Synonymous Form: whole *includes* part

Definition: The part is a component of the whole.

Note: There are a number of axioms of mereology that apply to the concept 'part *is part of* whole.' The following 3 axioms specify only that subset of those axioms that are needed by this specification. This subset is needed to define the partial ordering relationship among time intervals.

Note: Axiom of *reflexivity*: every part is part of itself.

Necessity: Each part *is part of* the part.

CLIF Axiom:
```
(forall ((part thing))
 ("part of" part part)
```

OCL Constraint:
```
context thing
inv: self.part-->exists(self)
```

Note: Axiom of *antisymmetry*: two distinct parts cannot be part of each other.

Necessity: If the part *is part of* the whole and the whole *is part of* the part then the part *is* the whole.

CLIF Axiom:
```
(forall ((part thing) (whole thing))
 (if (and ("part of" part whole)
          ("part of" whole part))
     (= part whole)))
```

OCL Constraint:
```
context thing
inv: self.whole-->exists(p |
                p.whole -->exists(self))
     implies self = self.whole
```

Note: Axiom of *transitivity*

Necessity: If the part *is part of* some whole and the whole *is part of* some $part_3$ then the part *is part of* $part_3$.

CLIF Axiom:
```
(forall ((part thing) (whole thing) (part3 thing))
  (if (and ("part of" part whole)
           ("part of" whole part3))
      ("part of" part part3)))
```

OCL Constraint:
```
context thing
inv: self.whole-->exists(whole |
      whole.whole-->exists(part3 |
       part3 implies self."part of"(part3)))
```

The combination of the reflexivity, antisymmetry, and transitivity axioms define a partial ordering among things that have the 'part *is part of* whole' relationship.

$part_1$ *overlaps* $part_2$

Synonymous Form: $part_2$ *overlaps* $part_1$

Definition: There exists a $part_3$ that *is part of* the $part_1$ and the $part_3$ *is part of* the $part_2$.

CLIF Definition:
```
(forall ((part1 thing) (part2 thing))
 (iff (overlaps part1 part2)
      (exists ((part3 thing))
       (and
        ("part of" part3 part1)
        ("part of" part3 part2)))))
```

OCL Definition:
```
context thing
inv: self.overlap-->forAll(part2 |
      self.part-->exists(part3 |
       part2.part-->exists(part3)))
```

Note: Two things overlap if they have some part in common.

Note: The synonymous form indicates that '$part_1$ *overlaps* $part_2$' is commutative, i.e., that the overlaps relationship is always mutual. This is obvious from the definition.

part *is a proper part of* whole

Definition: the part *is part of* the whole and the whole *is* not *part of* the part

CLIF Definition:
```
(forall ((whole thing) (part thing))
 (iff ("proper part " part whole)
      (and
```

```
        ("part of" part whole)
        (not ("part of" whole part)))))
```

OCL Definition:

```
context thing
inv: self."proper part"-->forall(pp |
    pp <> self)
```

Note: A proper part is a part that is not the whole.

Note: Axiom of *supplementation*: If a whole has a proper part, then it has more than one proper part.

Necessity: If a part$_1$ *is a proper part of* a whole then there exists a part$_2$ that *is a proper part of* the whole and part$_2$ *does* not *overlap* part$_1$.

CLIF Definition:

```
(forall ((part1 thing) (whole thing))
 (if ("proper part" part1 whole)
    (exists ((part2 thing))
     (and
      ("proper part" part2 whole)
      (not (overlaps part2 part1))))
```

OCL Definition:

```
context thing
inv: self."proper part"-->forAll(part1 |
    self."proper part"-->exists(part2 |
      not part2.overlaps(part1)))
```

Annex E - Formalizing English Tense and Aspect

(informative)

The normative clauses of this specification deal the semantics of time as used natural languages. This Annex describes how propositions that are given in English language syntax may be formulated using the Date-Time Vocabulary.

E.1 Syntax and Semantics of Time

Many natural languages have built-in syntactical mechanisms for expressing when an action occurs relative to the time of utterance or writing, or relative to the occurrence of another event. They also have standard ways of indicating whether and when an action is progressing or is accomplished. These mechanisms include the use of affixes with verbs, called *tense*, and the use of auxiliary verbs together with the main verb of a clause, called *aspect*. Not all languages have the same set of these kinds of mechanisms.

The terms ‘tense’ and ‘modal’ are used with somewhat different connotations when referring to syntax or semantics. In syntactic theory, ‘tense’ refers to different verb forms used to denote different times: past, present, or future. The term ‘tense’ can also be used to refer to the semantics of a temporal expression: the past tense, the present tense, the future tense. All languages incorporate mechanisms to express such semantics, but different languages have different syntactical mechanisms for doing so. Confusion sometimes arises in English, which has verb forms only for present tense and past tense. Consequently, it is common for some authors to say that English has only two tenses, past and present, and no future tense. At the same time, it is often said that the future tense in English is expressed using the auxiliary verb ‘will’. In this annex, ‘tense’ refers to verb forms that express past or present time, and ‘aspect’ to the use of auxiliaries to generate different senses of past, present, and future time. In the normative clauses of the specification, ‘tense’ refers to the semantics of past, present, or future time, without regard to the syntactical mechanisms employed to express time in any language.

The term ‘modal’ can be confused with ‘modality.’ In this annex, ‘modal’ is a grammatical term that refers to a modal verb (*see ‘modal’* below). ‘Modality’ is a logical term, used in SBVR, to refer to the mood of a proposition as involving the affirmation of either possibility, impossibility, necessity, contingency, obligation, or permission. SBVR includes a modal logic for these modalities, including modal formulae and modal negation rules. This specification does not provide a temporal logic for the temporal modality, rather temporal concepts are handled by the introduction of first order concepts and fact types defined in the normative clauses of this specification. No temporal logical operations are introduced in this specification. Negation of propositions involving time is treated conventionally as logical negation as specified in SBVR.

English syntax involving modal auxiliary verbs serves to denote both the tense and the logical mood of a proposition. The meaning depends on the particular auxiliary verbs used. A temporal connotation can be associated with each auxiliary verb, such that auxiliary verbs carry both a temporal connotation and a mood. The following table gives some examples.

Table E.1 - Modalities for Auxiliary Verbs

Auxiliary Verb	Time Frame	Modality
can	present	possibility
can not	present	impossibility
could	past	possibility
do not	future	negation
does not	present	negation
did not	past	negation
may	present	permission
might	past*, future	possibility
must	past*, future	obligation
need	always	necessity
shall	future	necessity
should	past*, future	contingency
used	past	--
will	future	--
would	past	--
	* with *have*	

Logical negation can be indicated by using *not* with an auxiliary verb; only a few examples are shown. *Always, never,* or *not ever* can be used with some modal auxiliary verbs to indicate *at all times*, or *not at any time*, as the case may be. Some words that serve as auxiliary verbs can have other grammatical roles as well. Time frame and modality can be expressed by means other than auxiliary verbs; this annex focuses on the behavior of English verbs in referring to time.

E.2 Organization of This Annex

This specification includes fact types that accurately capture the meaning of relationships between states of affairs and time, but the fact type forms needed for precise definition are not idiomatic. This annex describes a way to accommodate idiomatic English expressions involving time, giving rules for mapping such expressions to concepts provided in this specification preparatory to creating closed logical formulations of the idiomatic expression. This treatment is informative, not normative; other approaches are possible. It is extensive but not exhaustive; the most common cases are treated, but not all possibilities. A formal grammar of the tense and aspect in English is provided, followed by a general algorithm for performing the syntax-to-semantics transformations for the twelve grammatical categories. Finally, a table of specific cases of the use of tense and aspect in English is provided.

This annex only describes formulations in which time is denoted by verbs. Other temporal constructs, such as the use of literal duration values and time coordinates and expressions involving relationships between time periods, are not discussed here.

This annex effectively extends the modal operations described in SBVR Annex F The RuleSpeak® Business Rule Notation, to include time, but stops short of being a full treatment of temporal modality.

E.3 Definitions

The following definitions are excerpted from Sag, Wasow, and Bender, *Syntactic Theory, Second Edition*, Stanford University, Center for the Study of Language and Information (2003), Glossary.

tense Finite verbs come in different form depending on the time they denote; these forms are called 'tenses'. English has present and past tense, exemplified by the present tense forms *walk* and *walks*, and by the past tense form *walked*. Some languages also have future tenses, but English uses other means (e.g., the modal [q.v.] *will*) to express future time.

aspect Many language have special grammatical elements for locating in time the situation referred to. Among the temporal notions often expressed are whether situations are in process or completed and whether they occur repeatedly. These notions are often called 'aspect,' and words or affixes whose function is to express aspect are called 'aspectual markers.' *See also* perfective, progressive.

finite verb A finite verb is one that is marked for tense [q.v.] (present or past, in English).

modal The English verbs *can, could, may, might, must, shall, should, will,* and *would*, along with their negated forms (*can't,* etc.) are referred to as 'modals' or 'modal verbs.' They share the following properties: they function only as finite verbs [q.v.]; they exhibit auxiliary behavior (negation, inversion, contraction, and ellipsis); they take base VP [verb phrase] compliments; and they show no agreement [q.v.] (i.e., no third-person singular *–s* suffix). Some other languages have similar syntactically distinctive classes of words expressing necessity, possibility, obligation, and permission; these are also known as modals.

agreement In many languages, the form of certain elements can vary to indicate such properties such as person [referring to the speaker, the hearer, or third parties], number [referring to single entities or multiple entities], gender, etc. Often, these variations are marked with affixes. Some grammatical relationships between pairs of linguistic elements require they agree on these properties. In English, for example, present tense verbs are marked to indicate whether the subjects are third-person singular (with the suffix *–s*), and nouns indicate plurality (also with a suffix *–s*). The systematic covariation of the forms of the subject and verb is called 'subject-verb agreement'. Similarly, pronouns must agree with their antecedents in person, number, and (if third-person) gender.

perfective Many languages have special verb forms or constructions used to indicate that the event denoted by the verb is completed. These are referred to as 'perfective' (or just 'perfect') in aspect. The English perfective involves the combination of *have* with a past participle [q.v.], as in *The dog has eaten the cake. See also* aspect.

progressive Special verb forms or construction used to indicate that the event denoted by the verb is in progress are referred to as 'progressive' aspect. The English progressive involves combination of *be* with a present participle [q.v.], as in *The dog is eating the cake. See also* aspect.

participle Certain nonfinite verbs – usually ones that share some properties with adjectives – are referred to as 'participles.' English has three types of participles: present participles, which end in *–ing* and usually follow some form of *be*; past participles, which usually end in *–ed* or *–en* and follow some form of *have*; and passive participles, which look exactly like past participles but indicate the passive voice [q.v.]. The three participles of *eat* are illustrated in the following sentences:

(i) Termites are eating the house.
(ii) Termites have eaten the house.
(iii) The house was eaten by termites.

E.4 English Grammar of Tense and Aspect

English grammar for tense and aspect can be defined as follows, using Extended Backus Nauer Form notation (ISO/IEC 14977 Information technology – Syntactic metalanguage - Extended BNF). '::=" means 'is defined as.' Each '::=' statement is a

production rule. Each production rule is terminated by ';'. The order of the symbols on the right hand side of each production rule is significant, unless delimited by '|'. '|' means 'or', a choice. Brackets '[]' indicate the element is optional. Quoted words are literals. Comments are included between '(*' and '*)'.

S ::= NP AUX VP; (* S–sentence, NP–noun phrase, VP–verb phrase *)

AUX ::= [MODAL] [PERF] [PROG]; (* auxiliary verb *)

MODAL ::= 'can' | 'could' | 'may' | 'might' | 'must' | 'shall' | 'should' | 'used' | 'will' | 'would';

PERF ::= 'have' | 'has' | 'had'; (* perfective *)

PROG ::= 'is' | 'are' | 'was' | 'were' | 'be' | 'been'; (* progressive *)

Additional Rules for Auxiliaries (AUX)

1. Auxiliaries are optional.
2. Auxiliaries precede any non-auxiliary verb.
3. Auxiliaries determine the form of the following verb.
4. Auxiliaries can co-occur with each other, but only in a fixed order.
5. Auxiliaries of any given type cannot iterate.

The modals all indicate future time. They have the additional property of expressing necessity, possibility, obligation, or permission, as discussed in SBVR.

Not all combinations generated by the above grammar are valid English. Other rules apply, not given, such as subject-verb agreement. *Not*, *never, always,* or *not ever* can be used with some modals; these grammatical details are outside the scope of this annex, but the methods of this annex can be extended to include them. The table in E.6 gives a listing of grammatical constructs that appear regularly in English.

Reference: Sag, Wasow, and Bender (ibid.), pp.392-394.

E.5 Formulating Tense and Aspect

This section needs to be updated to reflect the term "situation model" as used in this specification, instead of "state of affairs."

The general approach used here to formulate a sentence involving tense or aspect is as follows:

1. Transform the sentence into a proposition based on the applicable fact type form in the conceptual schema, noting the original tense and aspect.
2. Identify the state of affairs that corresponds to the base proposition.
3. Restrict the state of affairs by instantiating one or more of the fact types defined in this specification involving states of affairs and time, as noted in 1.
4. Create closed logical formulations that mean the base proposition and its restrictions, as described in SBVR.

Transform to a base proposition

All propositions in SBVR are considered to be true or false when considered with respect to a given fact model. A proposition might be true when considered with respect to one fact model, and false when considered with respect to another. Each fact

model is taken to be a snapshot of the state of the universe of discourse at some time. The fact model is tantamount to a database, and the veracity of each proposition is based on the facts in the database at the time of the snapshot, which time may or may not be stated. This is standard SBVR.

Propositions in standard SBVR are expressed preferably in the simple present tense when finite verbs are used. Such propositions are considered untensed, as they apply to any fact model representing the state of the universe at the snapshot time of the fact model. Propositions involving non-finite verbs are also considered untensed in standard SBVR.

This specification includes the concept now, which is the current time, or present. When evaluating propositions using this specification, now is the snapshot time of the fact model with respect to which the propositions are being evaluated.

Transforming a proposition into an base form involves changing the verb to the tense of the applicable fact type in the conceptual schema, maintaining subject-verb agreement.

For example, the present perfect progressive sentence “Acme has been trading with Xycore” transforms to untensed “Acme trades with Xycore,” with the notation that the original is present perfect progressive. These sentences are both based on the fact type company$_1$ *trades with* company$_2$.

The guidance is generally not to encode tense or aspect into fact type forms unless the domain model specifically requires a particular tense or aspect for that fact type. Consider this example, “Six tasks have completed on May 5, 2010” may be based on the fact type “task *completed on* time point.” This fact type has an intransitive past tense verb. The conceptual schema has already restricted facts of this type to past or perfected. The example transforms to “Six tasks completed on May 5, 2010” with a notation that the original is present perfect. A different conceptual schema might include the fact type “task *completes on* time point” instead. The proposition then transforms to “Six tasks complete on May 5, 2010” with the same present perfect notation. The *“completes on”* fact type, unlike the *“completed on”* version, could be used for facts about future planned completions (*will complete on*). This illustrates that there is a certain economy in using simple-present fact type forms in domain models: every different tense and aspect variation of these sentences is based on the same fact type and transforms to the same untensed form.

Identify the state of affairs

The state of affairs of interest is the one that corresponds to the transformed sentence, the base form. This is an instance of the SBVR fact type meaning *corresponds to* thing, where the meaning is the base proposition and the thing is a state of affairs.

Restrict the state of affairs

The state of affairs is restricted by involving it in a role in an instance of appropriate fact type(s) from this specification. Which fact types to use depends on the tense and aspect of the original sentence, as noted at the time the base proposition was created. Create a fact instance of each of the appropriate fact types.

Create closed logical formulations

A closed logical formulation is created for the conjunction of the base proposition and the restricting facts. This constitutes the closed logical formulation of the original sentence.

E.6 Mapping Tense and Aspect to the Date-Time Vocabulary

This table is extensive but not exhaustive. Different modals can be substituted for ‘will,’ with other restrictions in the logical formulation (e.g., obligatory). In some of the examples, the ‘now’ time is apparently in the past, to accord with the history of James Joyce.

Table E.2 - Mapping Tense and Aspect to the Date-Time Vocabulary

MODAL	PERF sg/pl	PROG sg/pl	Verb Form	Grammatical Term	Example: person *writes* book	Date-Time Vocabulary Fact Type
			present	present simple	Joyce writes Ulysses.	None. This is the base state of affairs (s) **s: "Joyce writes Ulysses"**
			past	past simple	Joyce wrote Ulysses.	**s *is in the past***
used			infinitive	past simple	Joyce used to write Ulysses.	**s *is in the past***
will			present	future simple	Joyce will not write Ulysses.	**s *is in the future* and** **s *is not an actuality***
		is/are	present participle	present progressive	Joyce is writing Ulysses in 1919.	**s *holds within* 1919**
		was/were	present participle	past progressive, imperfective	Joyce was writing Ulysses in 1919.	**s *is in the past* and** **s *holds within* 1919**
will		be	present participle	future progressive	Joyce will not be writing Ulysses in 2012.	**s *is in the future* and** **s does not *hold within* 2012**
	has/ have		past participle	present perfective	Joyce has written Ulysses.	**s *is accomplished***
	had/had		past participle	past perfective, pluperfect	Joyce had written Ulysses by 1922.	**s *is in the past* and** **s *is accomplished* and** **s *occurs before* 1922**
will	have		past participle	future perfective	Joyce will have written Ulysses by 1922.	**s *is in the future* and** **s *is accomplished* and** **s *occurs before* 1922**
	has/ have	been	present participle	present perfect progressive	Joyce has been writing Ulysses in 1919.	**s *holds within* 1919 and** **s *is accomplished***
	had	been	present participle	pluperfect progressive	Joyce had been writing Ulysses in 1919.	**s *is in the past* and** **s *holds within* 1919 and** **s *is accomplished***
will	have	been	present participle	future perfect progressive	By the end of 1920, Joyce will have been writing Ulysses for 33 months.	**s *is in the future* and** **s *holds during* December 1920 –** **33 months**

Annex F - Simplified Syntax for Logical Formulations

(informative)

The table below is kindly provided by Don Baisley of Microsoft Corp. It describes a simplified syntax for showing SBVR logical formulations. This syntax is more compact and more readable than, but fully equivalent to, the syntax used in [SBVR]. As time permits, the various logical formulations given in this specification will be converted to use this syntax. The semantics intended by this syntax is exactly the semantics given in [SBVR].

Table F.1 - Simplified Syntax for Logical Formulations

	Syntax	SBVR Term
‹logical formulation›		
	‹fact type›(‹bindable target› [, ‹bindable target›]…)	atomic formulation
	‹noun concept›(‹bindable target›)	instantiation formulation
	necessary ‹logical formulation›	necessity formulation
	obligatory ‹logical formulation›	obligation formulation
	permissible ‹logical formulation›	permissibility formulation
	possible ‹logical formulation›	possibility formulation
	and ‹logical formulation› ‹logical formulation›	conjunction
	or ‹logical formulation› ‹logical formulation›	disjunction
	equivalent ‹logical formulation› ‹logical formulation›	equivalence
	xor ‹logical formulation› ‹logical formulation›	exclusive disjunction
	if ‹logical formulation› then ‹logical formulation›	implication
	not ‹logical formulation›	logical negation
	nand ‹logical formulation› ‹logical formulation›	nand formulation
	nor ‹logical formulation› ‹logical formulation›	nor formulation
	whether or not ‹logical formulation› ‹logical formulation›	whether-or-not formulation
	for each ‹variable introduction› ‹logical formulation›	universal quantification
	exists ‹variable introduction› [‹logical formulation›]	existential quantification
	exists ‹n›..* ‹variable introduction› [‹logical formulation›]	at-least-n quantification
	exists 0..‹n› ‹variable introduction› [‹logical formulation›]	at-most-n quantification
	exists ‹n›..‹n› ‹variable introduction› [‹logical formulation›]	exactly-n quantification
	exists ‹n›..‹n› ‹variable introduction› [‹logical formulation›]	numeric range quantification
	object(‹bindable target›) ‹logical formulation›	objectification
	aggregate(‹bindable target›) ‹projection›	aggregation formulation
	noun concept(‹bindable target›) ‹projection›	noun concept nominalization
	fact type(‹bindable target›) ‹projection›	fact type nominalization
	question(‹bindable target›) ‹projection›	question nominalization
	answer(‹bindable target›) ‹projection›	answer nominalization

Table F.1 - Simplified Syntax for Logical Formulations

	proposition(‹bindable target›) ‹logical formulation›	proposition nominalization
‹projection›	{ ‹variable introduction› [‹variable introduction›]… [● ‹variable introduction› [‹variable introduction›]…] [|‹logical formulation›] }	projection projection variable before ● auxiliary variable after ●
‹bindable target›	‹variable›	… *binds to* variable
	‹quoted literal text›	… *binds to* text
	‹individual concept›	… *binds to* individual concept
‹variable introduction›	‹variable› [: ‹concept›] [unitary] [where ‹logical formulation›]	variable
‹variable›	a letter followed by zero or more letters, digits or hyphens	variable
‹concept›	single-quoted signifier that represents a concept	concept
‹individual concept›	single-quoted signifier that represents an individual concept	individual concept
‹fact type›	single-quoted fact type form – placeholder order matches role order in bindings that follows	fact type
‹quoted literal text›	text in double quotes	text
‹n›	positive integer	cardinality

Indenting rule: Each ‹logical formulation› and ‹variable introduction› starts to the right of every syntactic construct that contains it.

Annex G - OWL/UML Diagrams

(informative)

This Annex contains UML diagrams of an OWL model of Annex D. The diagrams match the OWL ontologies included with this specification. The intent is to eventually extend these OWL ontologies to address the complete specification. The OWL ontologies, and these diagrams, were contributed by Elisa Kendall.

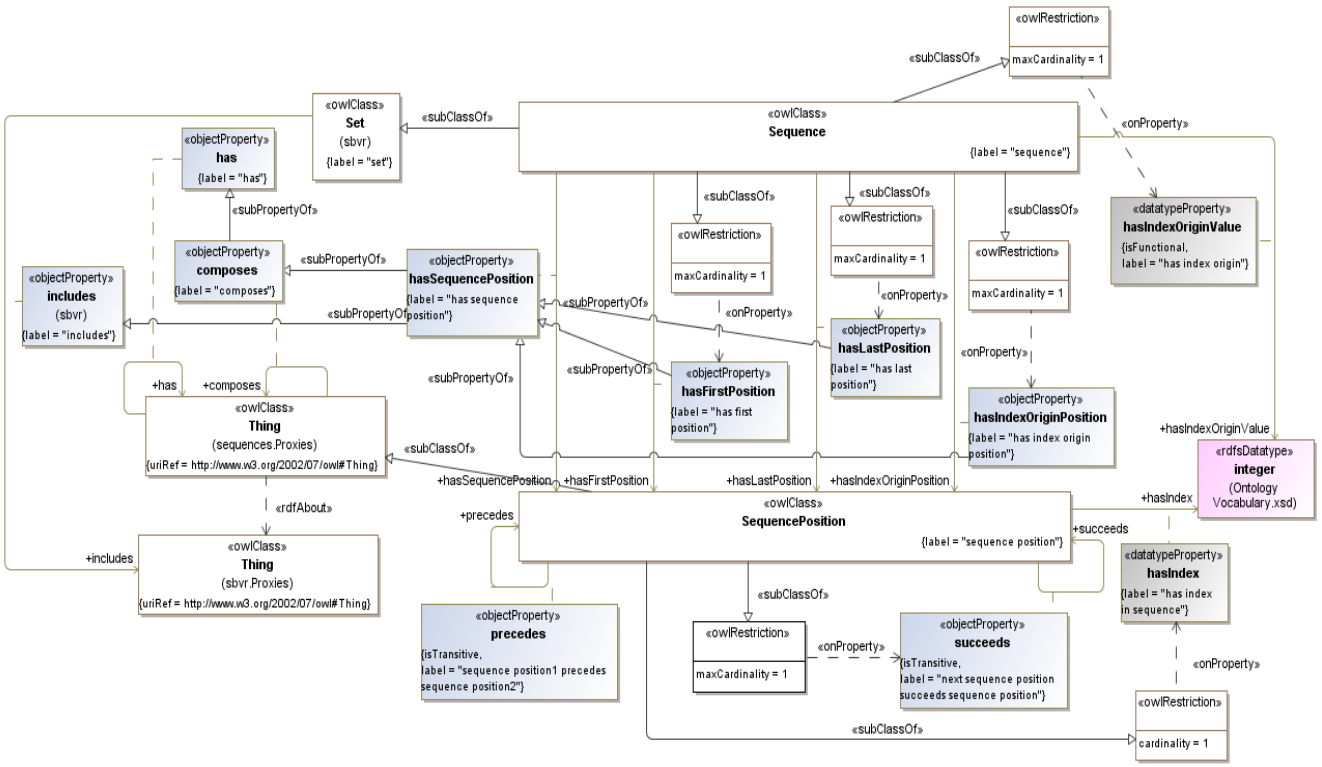

Figure G.1 - OWL/UML Model of Sequences

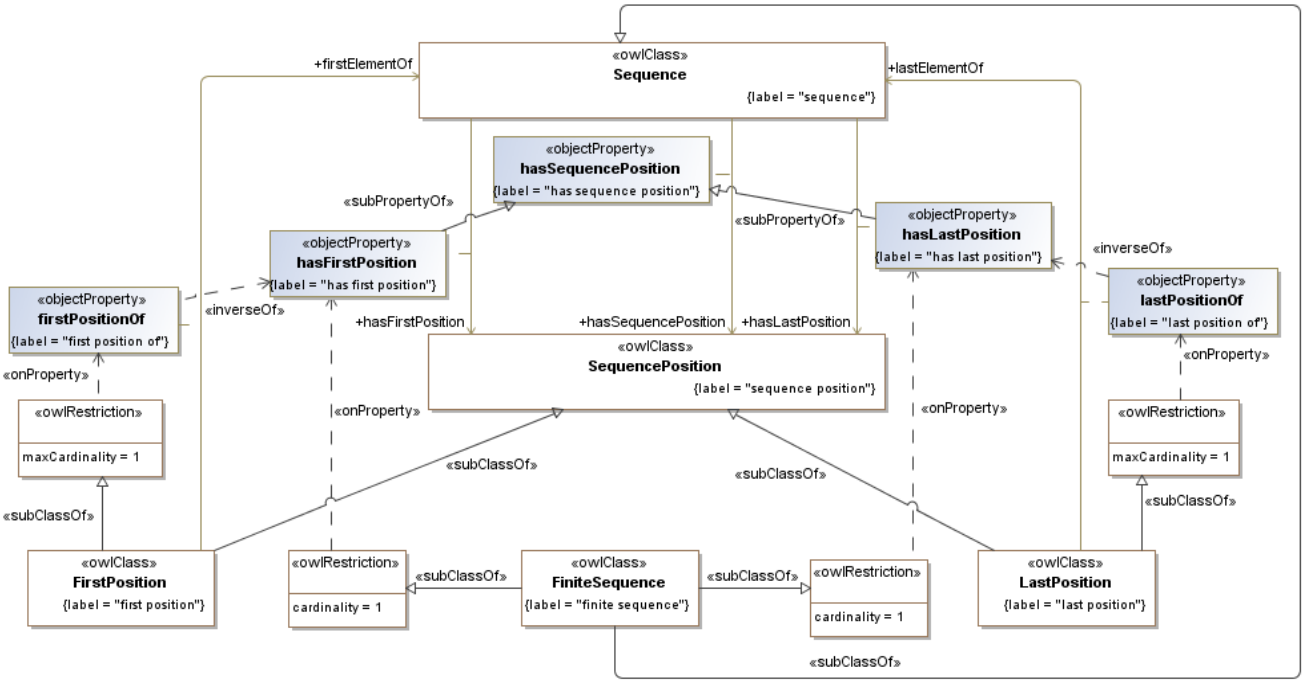

Figure G.2 - OWL/UML Model of "First Position" and "Last Position"

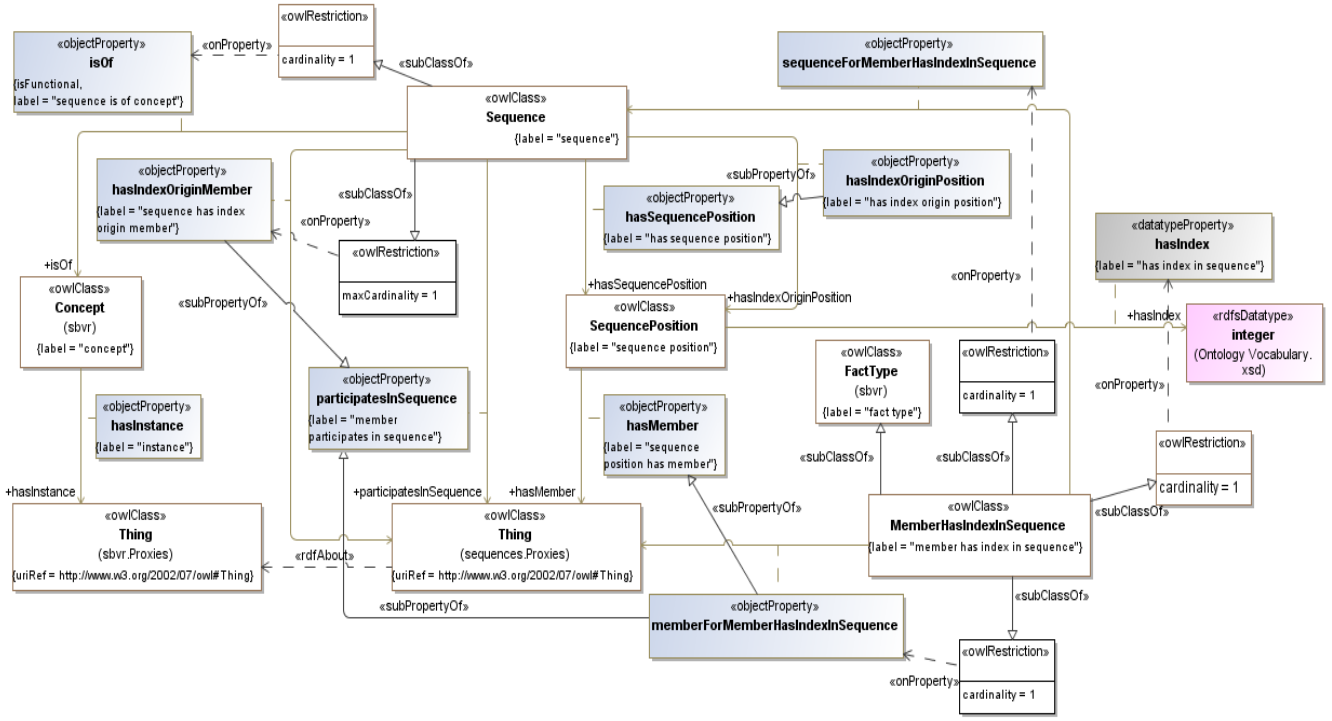

Figure G.3 - OWL/UML Model of Sequence Members

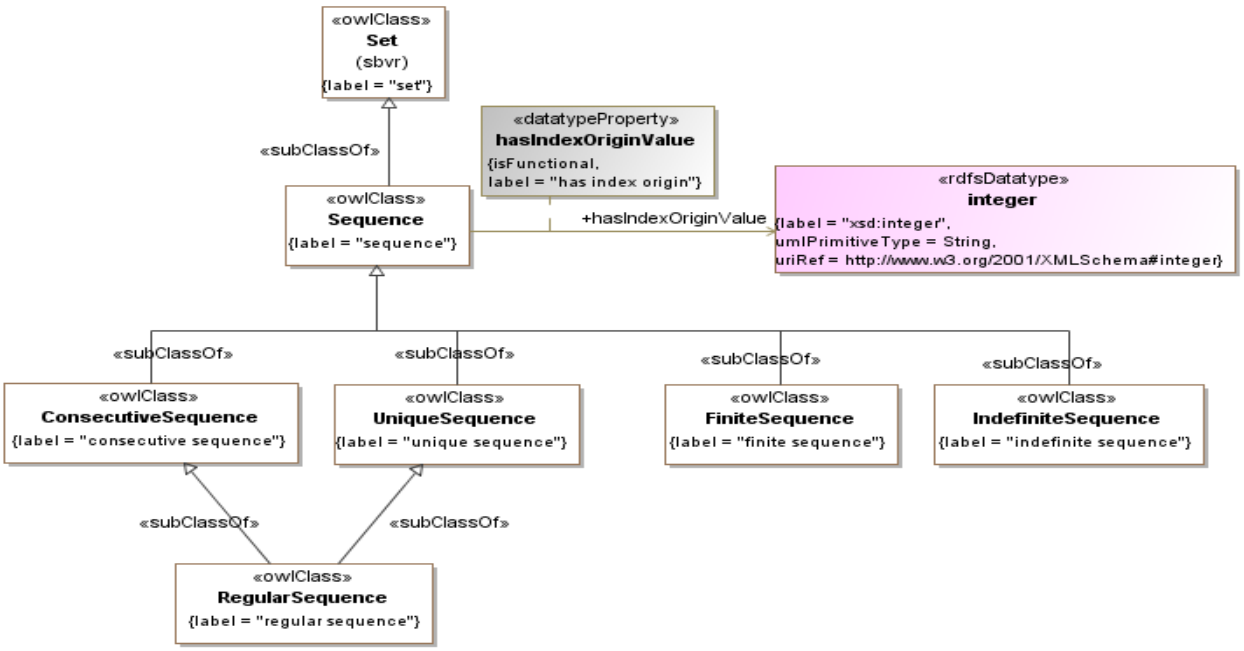

Figure G.4 - OWL/UML Model of Kinds of Sequences

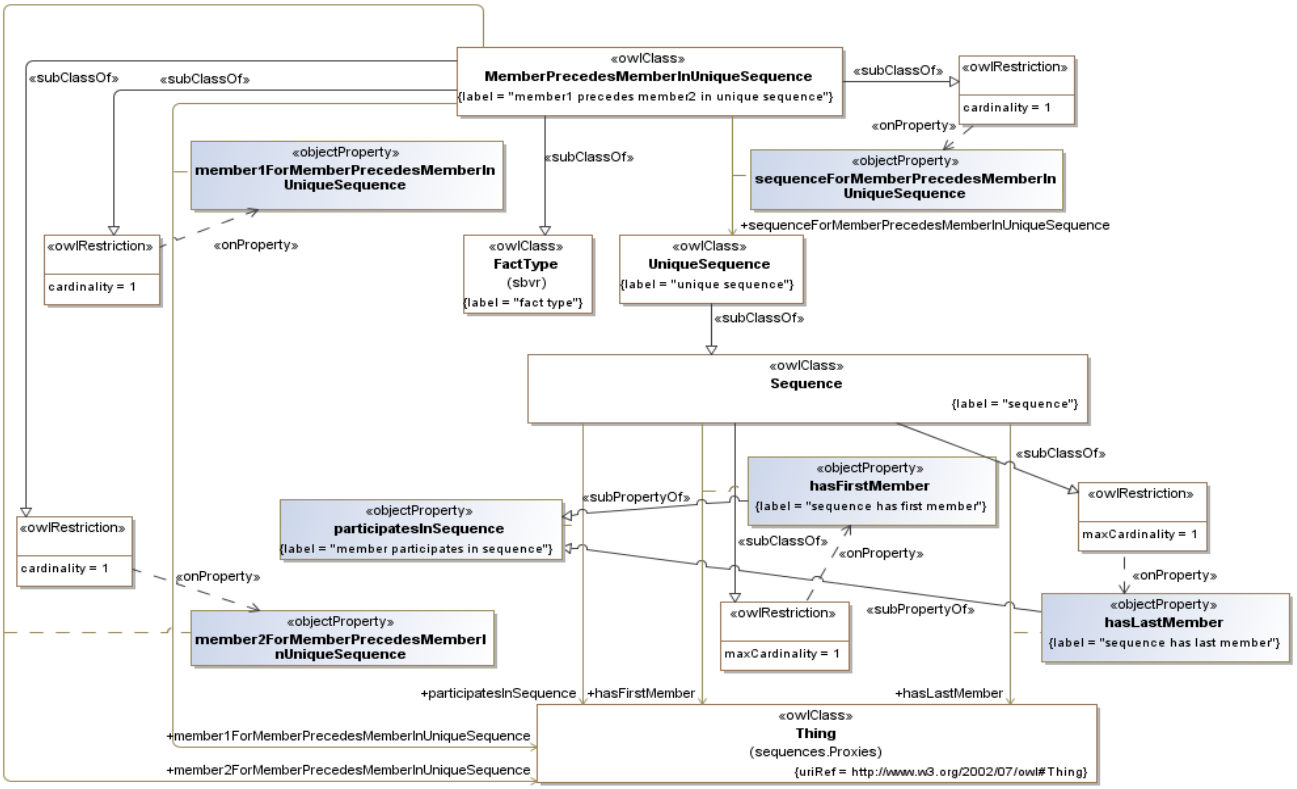

Figure G.5 - OWL/UML Model of "member 1 precedes member 2 in unique sequence"

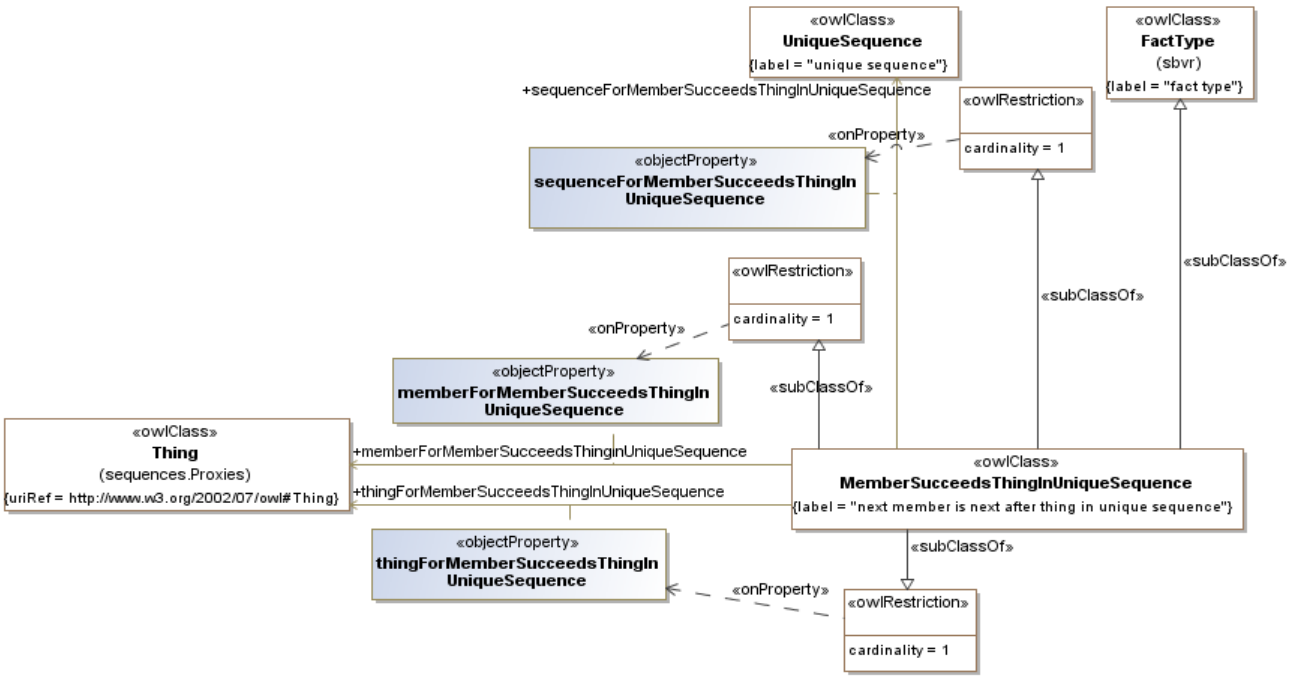

Figure G.6 - OWL/UML Model of "next member is next after thing in unique sequence"

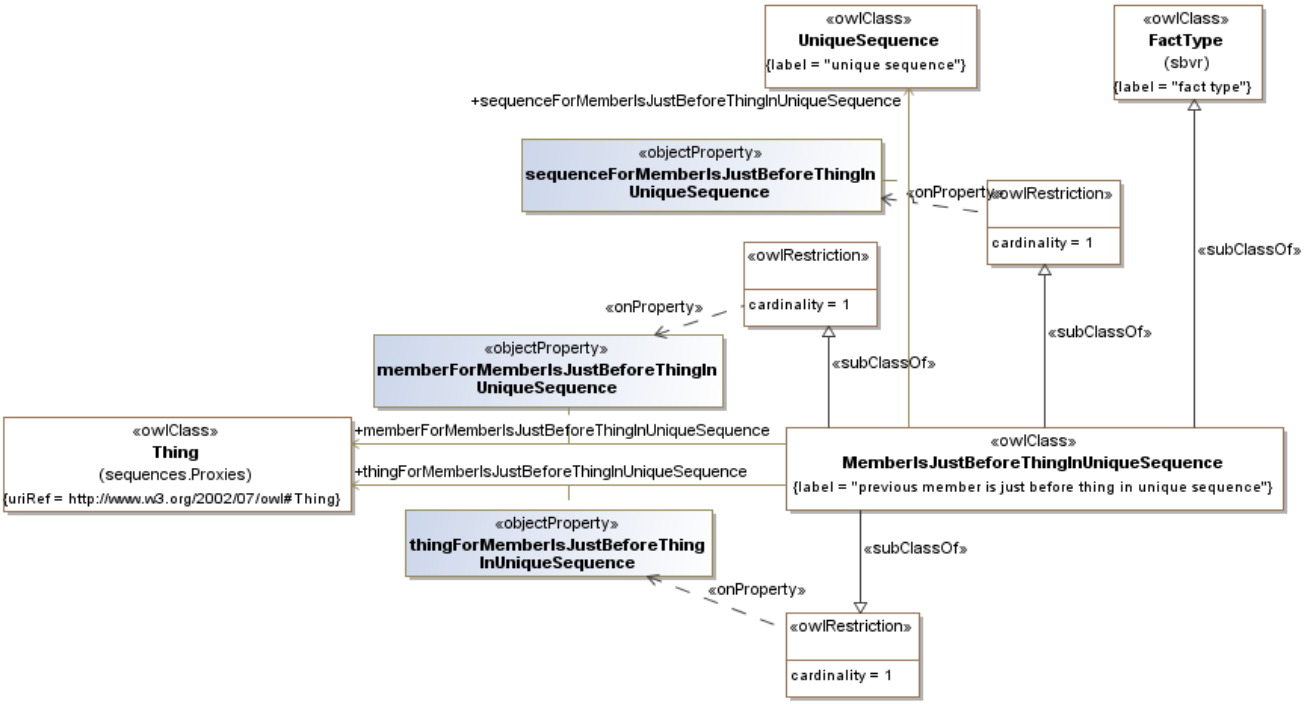

Figure G.7 - OWL/UML Model of "previous member is just before thing in unique sequence"

Annex H - Index of Business Designations

(informative)

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W

Y