Testbed-12 Aviation SBVR Engineering Report

Table of Contents

Publication Date: 2017-06-30

Approval Date: 2016-09-15

Posted Date: 2016-08-08

Reference number of this document: OGC 16-061

Reference URL for this document: http://www.opengis.net/doc/PER/t12-E004

Category: Public Engineering Report

Editor: Timo Thomas, Aleksandar Balaban

Title: Testbed-12 Aviation SBVR Engineering Report

Testbed-12 Aviation SBVR Engineering Report (16-061)

COPYRIGHT

WARNING

This document is an OGC Public Engineering Report created as a deliverable of an initiative from the OGC Innovation Program (formerly OGC Interoperability Program). It is not an OGC standard and not an official position of the OGC membership. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements. However, the discussions in this document could very well lead to the definition of an OGC Standard.

destroying the Intellectual Property together with all copies in any form. The license will also terminate if you fail to comply with any term or condition of this Agreement. Except as provided in the following sentence, no such termination of this license shall require the termination of any third party end-user sublicense to the Intellectual Property which is in force as of the date of notice of such termination. In addition, should the Intellectual Property, or the operation of the Intellectual Property, infringe, or in LICENSOR's sole opinion be likely to infringe, any patent, copyright, trademark or other right of a third party, you agree that LICENSOR, in its sole discretion, may terminate this license without any compensation or liability to you, your licensees or any other party. You agree upon termination of any kind to destroy or cause to be destroyed the Intellectual Property together with all copies in any form, whether held by you or by any third party.

Except as contained in this notice, the name of LICENSOR or of any other holder of a copyright in all or part of the Intellectual Property shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Intellectual Property without prior written authorization of LICENSOR or such copyright holder. LICENSOR is and shall at all times be the sole entity that may authorize you or any third party to use certification marks, trademarks or other special designations to indicate compliance with any LICENSOR standards or specifications.

This Agreement is governed by the laws of the Commonwealth of Massachusetts. The application to this Agreement of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded. In the event any provision of this Agreement shall be deemed unenforceable, void or invalid, such provision shall be modified so as to make it valid and enforceable, and as so modified the entire Agreement shall remain in full force and effect. No decision, action or inaction by LICENSOR shall be construed to be a waiver of any rights or remedies available to it.

None of the Intellectual Property or underlying information or technology may be downloaded or otherwise exported or reexported in violation of U.S. export laws and regulations. In addition, you are responsible for complying with any local laws in your jurisdiction which may impact your right to import, export or use the Intellectual Property, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

Abstract

This engineering report (ER) is a deliverable of the OGC Testbed 12. It advances previous work in the area of business rules for AIXM 5 based on SBVR. It evaluates the use of geo-spatial operators and constraints in SBVR, including a proof of concept for their automatic interpretation by software. It gives guidelines on how to deal with temporality aspects and how to extend the applicability of SBVR towards filtering expressions and it identifies limitations of the currently available vocabulary.

Business Value

In the context of AIXM, business rules ensure the consistency of the data. Syntactical correctness is enforced by the XML Schema, whereas semantical correctness is described using SBVR rules. This OGC document extends the applicability of SBVR towards business rules on the data content level and involving geo-spatial and temporal constraints.

What does this ER mean for the Working Group and OGC in general

This ER supports the overall objective of the OGC Aviation DWG to deliver recommendations on how to best use OGC standards in the context of SWIM / ATM information exchanges. The content of this ER is therefore expected to be used as appropriate by standardization organizations and/or governance bodies in charge of the development of ATM standards for SWIM.

Keywords

ogcdocs, testbed-12, SBVR, AIXM, business rules

Proposed OGC Working Group for Review and Approval

Aviation Domain Working Group

1. Introduction

1.1. Scope

This OGC document advances previous work in the area of business rules for AIXM 5 based on SBVR. It

- evaluates the use of geo-spatial operators and constraints in SBVR, including a proof of concept for their automatic interpretation by software,
- gives guidelines on how to deal with temporality aspects and how to extend the applicability of SBVR towards filtering expressions, and
- identifies limitations of the currently available vocabulary.

This OGC document is applicable to business rule definition in AIXM 5.

1.2. Document contributor contact points

All questions regarding this document should be directed to the editor or the contributors:

Table 1. Contacts

Name	Organization
Timo Thomas (Editor)	m-click.aero
Aleksandar Balaban	m-click.aero

1.3. Future Work

The following items were identified for consideration in future initiatives:

- extraction of a grammar for filtering expressions based on SBVR,
- formal definition of geometry literals for SBVR,
- introduction geometry construction functions in SBVR,
- introduction of named expressions in SBVR available for reference,
- separation of complex SBVR rules into multiple sentences to preserve the readability for humans, and
- evaluation of the use cases identified in Other Related Use Cases Referred to Future Work.

1.4. Foreword

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

2. References

The following documents are referenced in this document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. For undated references, the latest edition of the normative document referred to applies.

- AIXM 5.1 - Business Rules (data verification) - Using SBVR and Schematron V0.5 (2015)
- OGC 15-024r2, OGC® Testbed 11 Aviation - Guidance on Using Semantics of Business Vocabulary and Business Rules (SBVR) Engineering Report (2015)
- SESAR Joint Undertaking: Guidance on Writing AIRM Constraints, Edition 01.00.01 (2014)
- OMG: Semantics of Business Vocabulary and Rules (SBVR) 1.1 (2013)
- ISO/IEC 19757-3:2006 Information technology — Document Schema Definition Language (DSDL) — Part 3: Rule-based validation — Schematron (2006)

3. Terms and definitions

For the purposes of this report, the definitions specified in Clause 4 of the OWS Common Implementation Standard [OGC 06-121r9] apply. In addition, the following terms and definitions apply.

3.1. Abbreviated terms

AIXM	Aeronautical Information Exchange Model
API	Application Program Interface
ARP	Aerodrome Reference Point
ATM	Air Traffic Management
D-NOTAM	Digital NOTAM
DWG	Domain Working Group
ER	Engineering Report
ETOPS	**E**xtended-range **T**win-engine **O**perational **P**erformance **S**tandards
FIXM	Flight Information Exchange Model
GML	Geography Markup Language
JSON	JavaScript Object Notation
NOTAM	Notice to Airmen
PIB	Pre-Flight Information Bulletin
RFQ	Request for quotation
SBVR	Semantics of Business Vocabulary and Rules
SWIM	System Wide Information Management
WKT	Well Known Text
XML	eXtensible Markup Language
XPath	XML Path Language

4. Overview

One of the topics addressed by the OGC Testbed 12 Aviation thread was to advance the use of constraints represented using Semantics of Business Vocabulary and Business Rules (SBVR). AIXM business rules are expressed as SBVR constraints [5]. In Testbed 11 [1], the vocabulary was extended by geo-spatial terms. The task regarding SBVR in the Aviation thread was to apply those terms to capture rules that identify operational constraints based on aircraft characteristics and infrastructure characteristics, including the actual status.

In chapter Use Case Discussion, this document evaluates the example use cases given in the RFQ, and more taken from current publications in that topic or by evaluating AIXM. For each of the use cases, a detailed analysis is performed regarding:

- Is the existing SBVR vocabulary sufficient to cover the use case?
- Does the vocabulary need to be extended, and if yes, where?
- Are the rules machine-readable and still sufficiently intuitive to be understood by domain experts?

SBVR is designed to express rules about data in order to check it for consistency. However, the given uses cases in the RFQ suggest that it could also be used to identify relevant data. In chapter Filtering Data Using SBVR, the more general use case of filtering data using SBVR is covered.

The AIXM Temporality Model enhances each feature by another dimension: time. This aspect is covered in chapter Dealing with Temporality in SBVR.

5. Status Quo & New Requirements Statement

This section explains the status quo and the new requirements that have been addressed by this ER.

5.1. Status Quo

In the context of AIXM, business rules ensure the consistency of the data. Syntactical correctness is enforced by the XML Schema, whereas semantical correctness is described using SBVR rules. The SBVR standard [3] - Semantic Business Vocabulary and Rules - defines the basis for a formal and detailed natural language declarative description of a complex entity, such as AIXM. The formal vocabularies and rules can be interpreted and used by computer systems, and are human-friendly at the same time.

In [5], the vocabulary and so-called business facts were defined for AIXM, together with general recommendations on how to create AIXM business rules. An initial set of approximately 1400 rules is also provided there. In Testbed 11, [1] defined a vocabulary with geo-spatial terms for SBVR and developed a tool for automatic conversion of SBVR business rules to Schematron. As a result, around 60% of rule set could be automatically translated to Schematron, but without support for geo-spatial operators.

5.2. Requirements Statement

The RFQ states that this ER shall advance SBVR to:

- capture rules that identify operational constraints based on aircraft characteristics and infrastructure characteristics, including the actual status, and
- to apply the spatial operators introduced in OGC 15-024r2.

6. Solutions

This section explains the solutions that have been envisioned at the beginning of the testbed, experimented with during the testbed, and that have either been discarded, or implemented, or the decision has been deferred to future activities.

6.1. Targeted Solutions

A use-case driven approach was chosen to address the requirements. We identified a number of use cases that affected at least one requirement and categorized them. Not all of the identified use cases could be discussed in detail because of time restrictions. For the ones chosen, we applied the existing SBVR vocabulary to it, and answered the following questions:

- Is the existing SBVR vocabulary sufficient to cover the use case?
- If yes, are the rules machine-readable and still sufficiently intuitive to be understood by domain experts?
- If not, can the vocabulary be extended to reach that goal?

From the evaluation of the use cases, some more general issues were identified, which are discussed in separate chapters:

- Using SBVR for filtering data (see Filtering Data Using SBVR) and
- Dealing with temporality (see Dealing with Temporality in SBVR).

6.1.1. Use Cases Discussed in Detail

The following use cases were discussed in detail:

1. Relating the aircraft type to the possibility to operate on a given runway, taking into consideration:
   - landing distances (see Landing Distances and Runways),
   - risk of overshooting the last exit taxiway (see Risk of overshooting the last taxiway),
   - limitations in turning radius of the aircraft (see Limitations in turning radius of the aircraft), and
   - the purpose (or flight status): regular or emergency landing (covered as part of Landing Distances and Runways);
2. Identification of airports along the trajectory of a flight (see Use Case 2: Identification of airports along the trajectory of a flight); and
3. From [1]: validation of an airport's address against its geo-location (ARP) (see Use Case 3: Validation of an airport's address against its geo-location).

6.1.2. Other Related Use Cases Referred to Future Work

The following use cases were also identified but couldn't be discussed in detail because of time restrictions:

- validation of a 'Displaced Runway Threshold' D-NOTAM: check if new position of the `RunwayCentreLinePoint` is collinear to the other center line points of the same runway,
- validation of runways: every `RunwayCentrelinePoint` shall be geometrically within in the corresponding `RunwayElement` geometry,
- identification of D-NOTAMs relevant for a flight for PIB creation: a spatial search on a buffer around a planned route,
- consistency check on routes: `RouteSegment.length` shall be equal to the spatial distance between `RouteSegment.start` and `RouteSegment.end`, and
- plausibility check on aerodromes: the geometries of various physical features of an airport shall be within some reasonable distance (e.g. 10km) of the ARP: e.g. `AircraftStand`, `RunwayBlastpad`, `RunwayElement`, `TaxiwayElement` etc.

6.1.3. Additional Tasks Accomplished

Though not required by the RFQ, we provided a proof of concept that existing tools for automatic SBVR translation are extendable towards the support of geo-spatial operators. We successfully extended ShapeChange and m-click.aero’s AIXM Validator, which is described in Use Case 3: Validation of an airport’s address against its geo-location.

6.2. Recommendations

A summary of recommendations appears below. For details, refer to the following chapters. The recommendations partially overlap with section Future Work, as a complete definition of the suggested extensions was out of scope of this ER.

- extend SBVR by geometry construction functions like the spatial union of a set of geometries;
- extend SBVR by geometry literals based on WKT;
- introduce specialized operators to check advanced constraints like:
  - the smoothed radius of a composed curve,
  - the collinearity of a set of points, and
  - the evaluation of a logical expression defined through objects of type `ConditionCombination`.

7. Use Case Discussion

Many use cases in the aviation domain require the evaluation of spatial constraints. A spatial vocabulary for SBVR has been defined in [1] and further detailed in [2]. It is one objective of this ER to evaluate this spatial vocabulary on real use cases.

This chapter provides a detailed discussion of the use cases identified in Targeted Solutions.

7.1. Use Case 1: Relating the aircraft type to the possibility to operate on a given runway

As seen in Figure 1 and Figure 3, in AIXM, airports can bind their availability to a complex combination of constraints on the aircraft, flight type and weather conditions. We assume that those constraints cover all relevant conditions that affect the suitability of the airport for a specific flight, hence including landing distances, the risk of overshooting the last taxiway, limitations in turning radius of the aircraft and the purpose of the flight. That said, the aforementioned conditions could still be validated against the physical properties of the airport, or more precisely, its runways and taxiways.

In the following analysis we'll see that formulating a rule based on the airport's selection of permitted aircraft types is not straight-forward. In order to avoid making the discussion too abstract, we chose the rather simple validation of runway lengths first, and discuss the additional complexity of 'real' spatial constraints afterwards.

7.1.1. Landing Distances and Runways

The actual constraint is as follows: the aircraft's landing distance shall be lesser or equal than the runway length. A look at the AIXM reveals (more detail below) that the runway length is represented as a scalar value. Hence, there is no need for a spatial calculation based on the runway geometry which means this use case does not require the application of spatial constraints (the use case of checking the runway's geometry against the runway's length property is a different one and wasn't evaluated in this ER).

The figures below from the AIXM 5.1 dictionary show that airports (`AirportHeliport`) are linked to `AircraftCharacteristic` and `FlightCharacteristic` objects. The `AircraftCharacteristic` object describes a type of aircraft. However, it does not contain landing distances directly. We assume that it can be derived or calculated from the properties `aircraftLandingCategory`, `speed` and `weight`, and also depends on whether it is a regular landing or an emergency landing. The latter information is described in objects of type `FlightCharacteristic` (property `status`). How the calculation can be done is out of scope here. For the purpose of consistency checks, we assume that the suitability for emergency landings is not based on runway length only, but may include obstacles and the topology around the runway. Because of this, we will formulate checks for regular landings only.

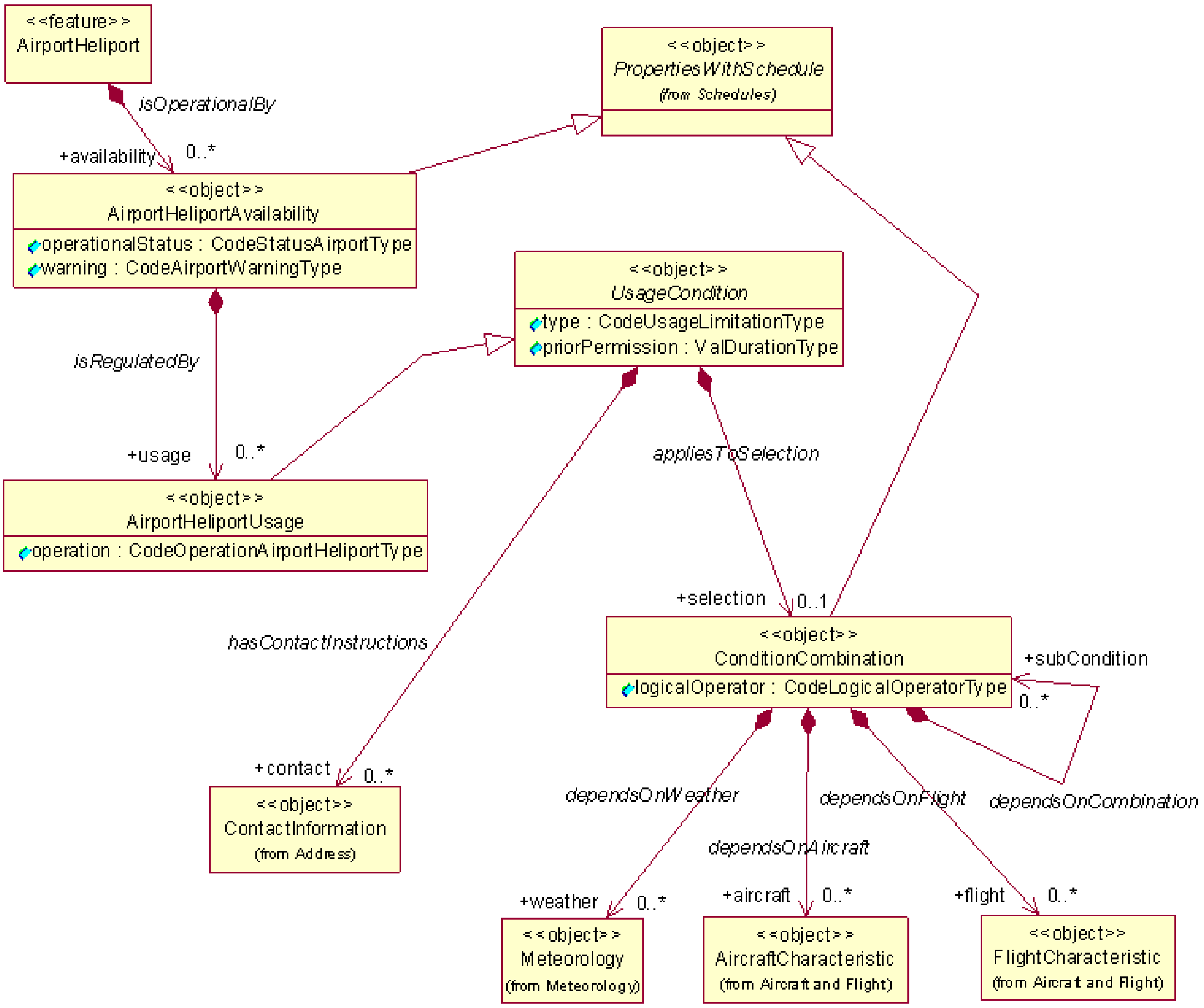


Figure 1. AIXM 5.1.1 AirportHeliport Availability Class Diagram

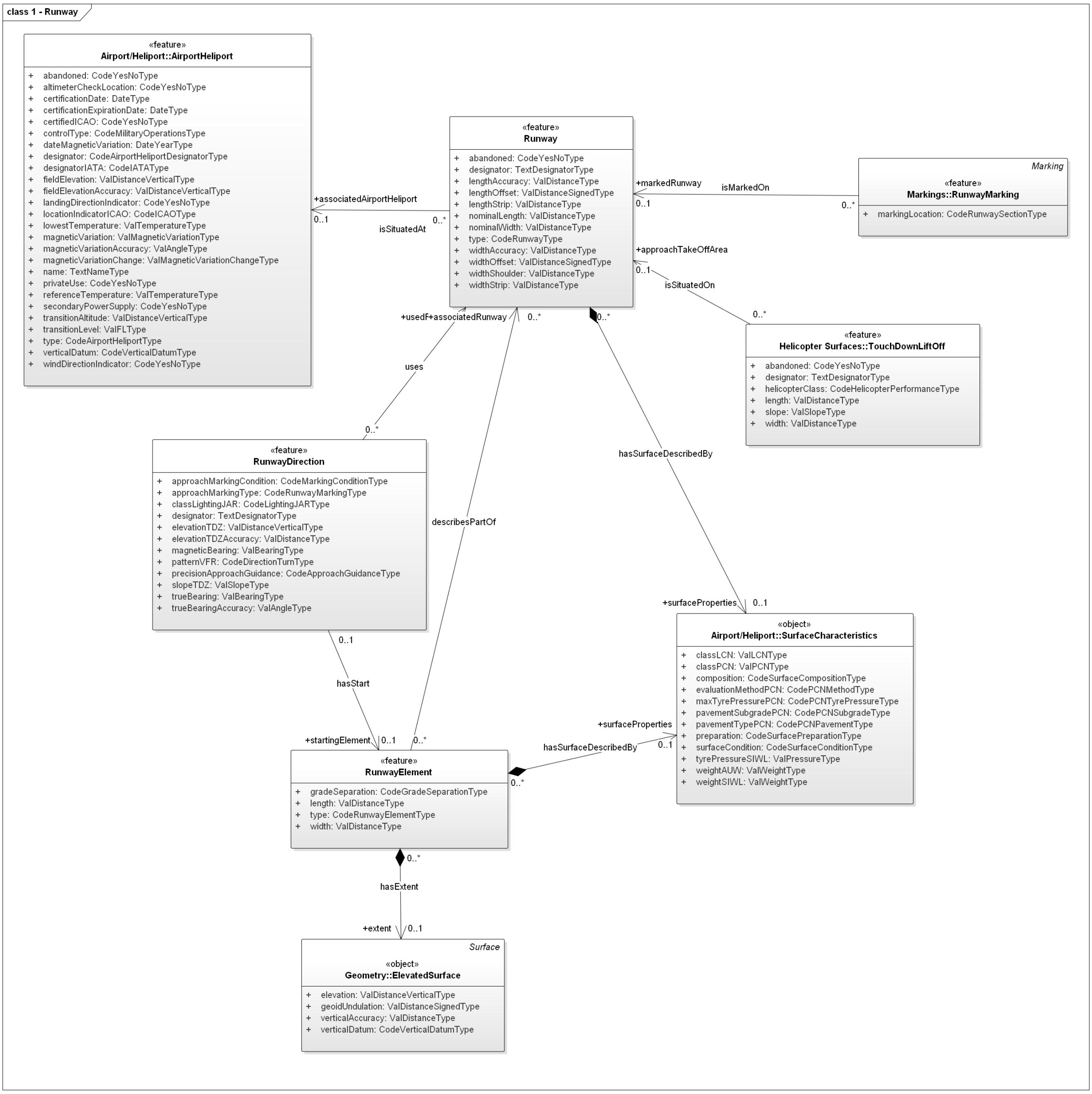

class 1 - Runway
«feature»
Airport/Heliport::AirportHeliport
+ abandoned: CodeYesNoType
+ altimeterCheckLocation: CodeYesNoType
+ certificationDate: DateType
+ certificationExpirationDate: DateType
+ certifiedICAO: CodeYesNoType
+ controlType: CodeMilitaryOperationsType
+ dateMagneticVariation: DateYearType
+ designator: CodeAirportHeliportDesignatorType
+ designatorIATA: CodeIATAType
+ fieldElevation: ValDistanceVerticalType
+ fieldElevationAccuracy: ValDistanceVerticalType
+ landingDirectionIndicator: CodeYesNoType
+ locationIndicatorICAO: CodeICAOType
+ lowestTemperature: ValTemperatureType
+ magneticVariation: ValMagneticVariationType
+ magneticVariationAccuracy: ValAngleType
+ magneticVariationChange: ValMagneticVariationChangeType
+ name: TextNameType
+ privateUse: CodeYesNoType
+ referenceTemperature: ValTemperatureType
+ secondaryPowerSupply: CodeYesNoType
+ transitionAltitude: ValDistanceVerticalType
+ transitionLevel: ValFLType
+ type: CodeAirportHeliportType
+ verticalDatum: CodeVerticalDatumType
+ windDirectionIndicator: CodeYesNoType
+associatedAirportHeliport
0..1
isSituatedAt
0..*
«feature»
Runway
+ abandoned: CodeYesNoType
+ designator: TextDesignatorType
+ lengthAccuracy: ValDistanceType
+ lengthOffset: ValDistanceSignedType
+ lengthStrip: ValDistanceType
+ nominalLength: ValDistanceType
+ nominalWidth: ValDistanceType
+ type: CodeRunwayType
+ widthAccuracy: ValDistanceType
+ widthOffset: ValDistanceSignedType
+ widthShoulder: ValDistanceType
+ widthStrip: ValDistanceType
+markedRunway
0..1
isMarkedOn
0..*
Marking
«feature»
Markings::RunwayMarking
+ markingLocation: CodeRunwaySectionType
+approachTakeOffArea
0..1
isSituatedOn
0..*
«feature»
Helicopter Surfaces::TouchDownLiftOff
+ abandoned: CodeYesNoType
+ designator: TextDesignatorType
+ helicopterClass: CodeHelicopterPerformanceType
+ length: ValDistanceType
+ slope: ValSlopeType
+ width: ValDistanceType
+usedF+associatedRunway
0..*
0..*
uses
0..*
hasSurfaceDescribedBy
«feature»
RunwayDirection
+ approachMarkingCondition: CodeMarkingConditionType
+ approachMarkingType: CodeRunwayMarkingType
+ classLightingJAR: CodeLightingJARType
+ designator: TextDesignatorType
+ elevationTDZ: ValDistanceVerticalType
+ elevationTDZAccuracy: ValDistanceType
+ magneticBearing: ValBearingType
+ patternVFR: CodeDirectionTurnType
+ precisionApproachGuidance: CodeApproachGuidanceType
+ slopeTDZ: ValSlopeType
+ trueBearing: ValBearingType
+ trueBearingAccuracy: ValAngleType
describesPartOf
+surfaceProperties
0..1
«object»
Airport/Heliport::SurfaceCharacteristics
+ classLCN: ValLCNType
+ classPCN: ValPCNType
+ composition: CodeSurfaceCompositionType
+ evaluationMethodPCN: CodePCNMethodType
+ maxTyrePressurePCN: CodePCNTyrePressureType
+ pavementSubgradePCN: CodePCNSubgradeType
+ pavementTypePCN: CodePCNPavementType
+ preparation: CodeSurfacePreparationType
+ surfaceCondition: CodeSurfaceConditionType
+ tyrePressureSIWL: ValPressureType
+ weightAUW: ValWeightType
+ weightSIWL: ValWeightType
0..1
hasStart
+startingElement
0..1
0..*
+surfaceProperties
hasSurfaceDescribedBy
0..1
0..*
«feature»
RunwayElement
+ gradeSeparation: CodeGradeSeparationType
+ length: ValDistanceType
+ type: CodeRunwayElementType
+ width: ValDistanceType
0..*
hasExtent
+extent
0..1
Surface
«object»
Geometry::ElevatedSurface
+ elevation: ValDistanceVerticalType
+ geoidUndulation: ValDistanceSignedType
+ verticalAccuracy: ValDistanceType
+ verticalDatum: CodeVerticalDatumType

Figure 2. AIXM 5.1.1 Runway Class Diagram

The class diagram about Runway and RunwayElement show that the length of a runway is contained in RunwayElement as a scalar value. That means this check can be expressed without the extended spatial operators from [1], but with as many SBVR rules as there are values of CodeAircraftCategoryType, because for each aircraft category a different minimum required length applies.

We will begin with a rough outline of the rules and then refine them to obtain a valid SBVR rule. A first version is:

```
It is prohibited that an AirportHeliport "that is available for a regular landing of
aircraft category X" "has" not at-least-one Runway with length greater-or-equal-to Y
{M}
```

where X denotes an aircraft category ('A' through 'E'), and Y suitable length values (Please note that [5] recommends use the term 'It is prohibited that' in favor of 'It is obligatory that').

The marked parts are inexact and/or not supported yet by the existing SBVR. We will discuss and refine them now:

- "has ... Runway": the association between `AirportHeliport` and `Runway` is named `isSituatedAt`, not 'has/have' and is oriented from runways to airports. However, the rule is based on the airport, and we cannot change this (the rule is about checking the validity of an airport, not a runway). We could not find a solution for this problem in [1] and [2] and assume that it is currently not supported. A possible verbalization is

"[AirportHeliport] has Runway that isSituatedAt it"

where "it" refers to the aforementioned object (here: `AirportHeliport`). Further evaluation of this has to be deferred to future activities.

"is available for regular landing of aircraft category X"

Again, this expression is not compatible with existing SBVR vocabulary as it does not refer to the elements in AIXM. The general availability is expressed through `AirportHeliportUsage.operation`, which is linked to `AirportHeliport` through the associations 'isOperationalBy (AirportHeliportAvailability)' and 'isRegulatedBy (AirportHeliportUsage)'. Further, `AirportHeliportUsage.type` is expressed if the described usage is permitted or forbidden (and more). Now we have:

```
It is prohibited that an AirportHeliport with
availability.AirportHeliportAvailability.operationalStatus equal-to 'NORMAL' and
[...](1) and
availability.AirportHeliportAvailability.usage.AirportHeliportUsage.operation equal-to
('ALL', 'LANDING') and
availability.AirportHeliportAvailability.usage.AirportHeliportUsage.type equal-to
('PERMIT','RESERV') [...](2) with AircraftCharacteristic.aircraftLandingCategory
equal-to X and FlightCharacteristic.status equal-to 'ALL' has not at-least-one Runway
that isSituatedAt it with length greater-or-equal-to Y {M}
```

The first ellipsis "[...](1)" stands for the part where additional constraints on the `PropertiesWithSchedule` object have to be set from which `AirportHeliportAvailability` is derived. This falls in the category 'dealing with the current state' in SBVR, which is discussed in a separate chapter (Chapter 10). For the further analysis we assume that the suggested solution presented there is applied, which moves the evaluation of the current state outside the SBVR rule.

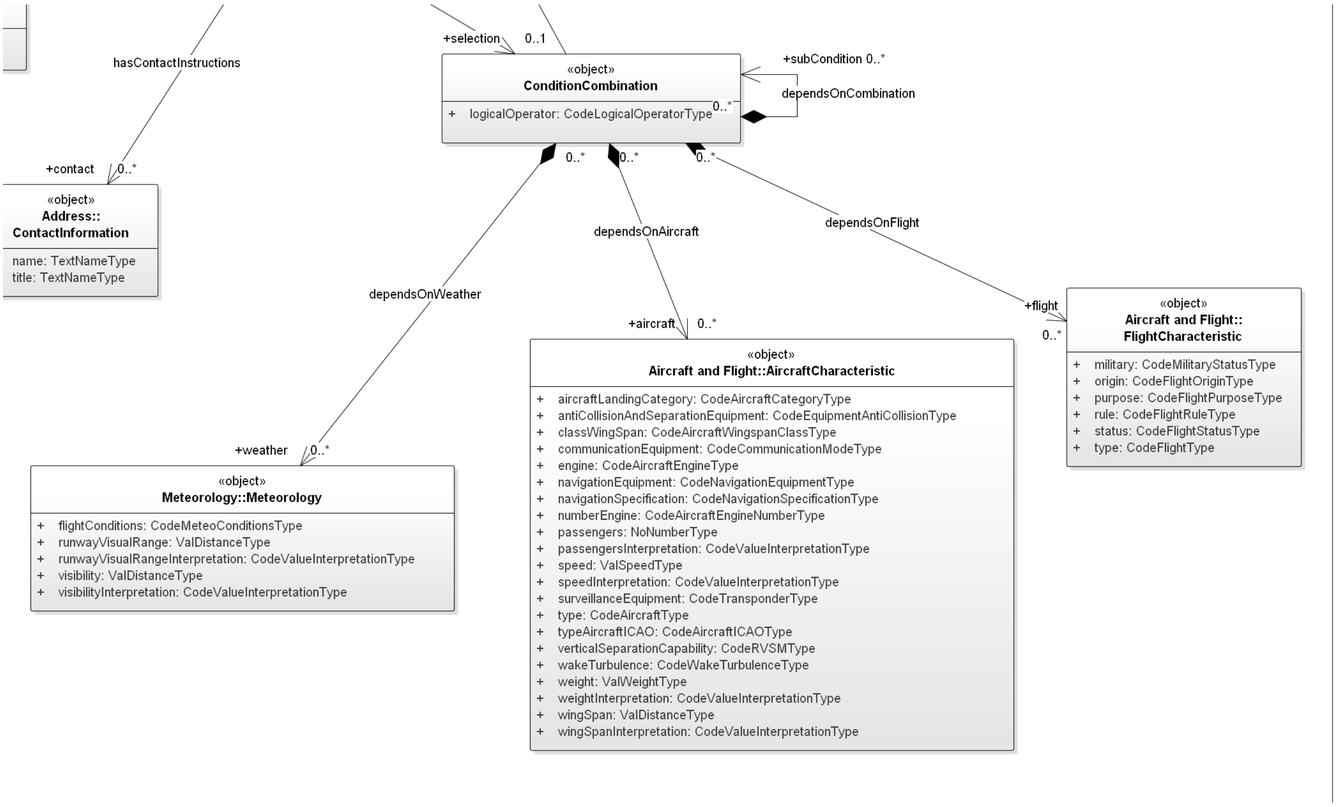

hasContactInstructions
+selection
0..1
«object»
ConditionCombination
+ logicalOperator: CodeLogicalOperatorType
+subCondition 0..*
dependsOnCombination
0..*
+contact
0..*
«object»
Address::
ContactInformation
name: TextNameType
title: TextNameType
0..*
0..*
0..*
dependsOnAircraft
dependsOnFlight
dependsOnWeather
+flight
0..*
«object»
Aircraft and Flight::
FlightCharacteristic
+ military: CodeMilitaryStatusType
+ origin: CodeFlightOriginType
+ purpose: CodeFlightPurposeType
+ rule: CodeFlightRuleType
+ status: CodeFlightStatusType
+ type: CodeFlightType
+aircraft
0..*
«object»
Aircraft and Flight::AircraftCharacteristic
+ aircraftLandingCategory: CodeAircraftCategoryType
+ antiCollisionAndSeparationEquipment: CodeEquipmentAntiCollisionType
+ classWingSpan: CodeAircraftWingspanClassType
+ communicationEquipment: CodeCommunicationModeType
+ engine: CodeAircraftEngineType
+ navigationEquipment: CodeNavigationEquipmentType
+ navigationSpecification: CodeNavigationSpecificationType
+ numberEngine: CodeAircraftEngineNumberType
+ passengers: NoNumberType
+ passengersInterpretation: CodeValueInterpretationType
+ speed: ValSpeedType
+ speedInterpretation: CodeValueInterpretationType
+ surveillanceEquipment: CodeTransponderType
+ type: CodeAircraftType
+ typeAircraftICAO: CodeAircraftICAOType
+ verticalSeparationCapability: CodeRVSMType
+ wakeTurbulence: CodeWakeTurbulenceType
+ weight: ValWeightType
+ weightInterpretation: CodeValueInterpretationType
+ wingSpan: ValDistanceType
+ wingSpanInterpretation: CodeValueInterpretationType
+weather
0..*
«object»
Meteorology::Meteorology
+ flightConditions: CodeMeteoConditionsType
+ runwayVisualRange: ValDistanceType
+ runwayVisualRangeInterpretation: CodeValueInterpretationType
+ visibility: ValDistanceType
+ visibilityInterpretation: CodeValueInterpretationType

Figure 3. AirportHeliport Availability ConditionCombination Detail

The second ellipsis "[…](2)" stands for the part where the `AircraftCharacteristic` and `FlightCharacteristic` objects are identified. These objects are linked to `AirportHeliportUsage` by an object of type `ConditionCombination`, with which simple logical expressions can be built. As can be seen in Figure 3, the logical expressions are combinations of `AircraftCharacteristic`, `FlightCharacteristic` and `Meteorology`, which are linked through boolean operators ("and", "or", "not" etc.). We need to express the constraint "logical expression (`ConditionCombination`) matches `AircraftCharacteristic` with `aircraftLandingCategory Y` and `FlightCharacteristic` with status `'ALL'`". It is obvious that it is not feasible to formulate this with first order logical connectives only. What is needed is a new predicate to express that (e.g. "matches").

The refined rule is:

```
It is prohibited that an AirportHeliport with
availability.AirportHeliportAvailability.operationalStatus equal-to 'NORMAL' and
availability.AirportHeliportAvailability.usage.AirportHeliportUsage.operation equal-to
('ALL', 'LANDING') and
availability.AirportHeliportAvailability.usage.AirportHeliportUsage.type equal-to
('PERMIT','RESERV') and
availability.AirportHeliportAvailability.usage.AirportHeliportUsage.selection.Conditio
nCombination matches (AircraftCharacteristic with aircraftLandingCategory equal-to X
and FlightCharacteristic with status equal-to 'ALL') has not at-least-one Runway that
isSituatedAt it with length greater-or-equal-to Y {M}
```

Discussion:

In order to formulate this rule, SBVR for AIXM needs

- to support resolving backward associations
- additional predicates for expressing complex constraints, including new grammar constructs (here: the content in parentheses after `matches`, which is an expression)
- to evaluate the current state outside the rule

However, the example shows that the resulting rule is very complex and hard to be read by humans. The main causes for this are

- AIXM-SBVR rules have to be formulated in one single sentence
- SBVR does not support variables that could be used to name objects for later reference
- the underlying model (AIXM) is very complex

Furthermore, we observe that there are ambiguities in the rules that have to be resolved. If two constraints, formulated on properties that are accessed through the dot operator ("."), share an ancestor path, and that ancestor path contains multiplicities greater than one, it is unclear whether both constraints refer to the same or different ancestor. In the rule above, this problem is visible in the expression:

```
availability.AirportHeliportAvailability.usage.AirportHeliportUsage.operation equal-to ('ALL', 'LANDING') and availability.AirportHeliportAvailability.usage.AirportHeliportUsage.type equal-to ('PERMIT','RESERV')
```

Multiple `AirportHeliportUsage` objects can be associated with `AirportHeliportAvailability`. Here, the constraint intends to express that at least one AirportHeliportUsage.usage object exists that **at the same time** fulfills `operation equal-to ('ALL', 'LANDING')` and `type equal-to ('PERMIT','RESERV')`. But it could also be read in a way that it only requires that one usage fulfills `operation equal-to ('ALL', 'LANDING')` and another one `type equal-to ('PERMIT','RESERV')`. This ambiguity can also be found in some rules of [5]. A possible solution is the introduction of variables.

7.1.2. Risk of overshooting the last taxiway

For this use case in a validation scenario, the constraint on the runway that was formulated in Landing Distances and Runways has to change, but the remaining constraints on the airport stay the same. The objective is to check whether the risk of overshooting the last taxiway is acceptable for the permitted aircraft types and flight status (regular or emergency landing). We assume that this risk is directly related to the distance of the last taxiway from the start of its runway.

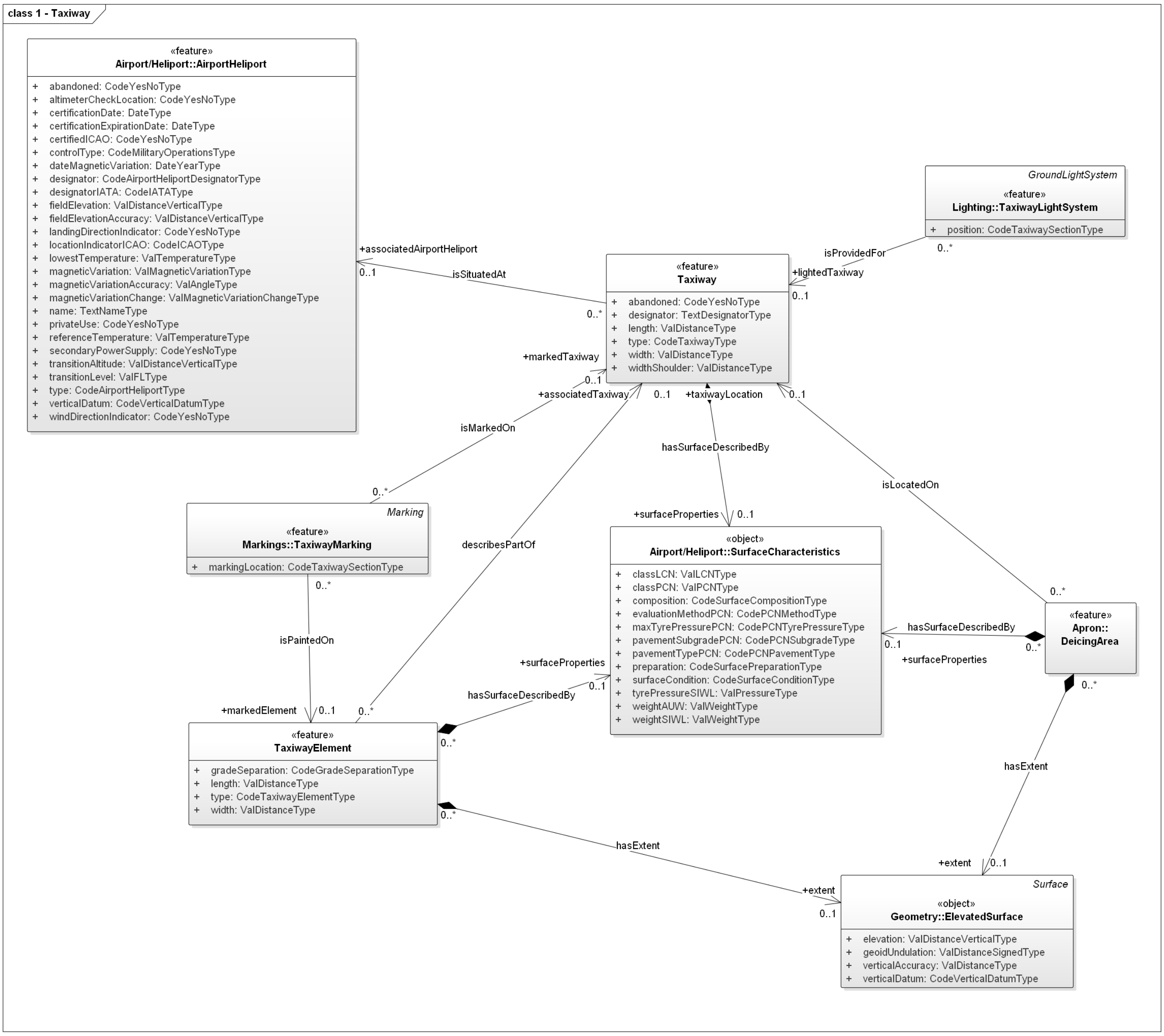

class 1 - Taxiway
«feature»
Airport/Heliport::AirportHeliport
+ abandoned: CodeYesNoType
+ altimeterCheckLocation: CodeYesNoType
+ certificationDate: DateType
+ certificationExpirationDate: DateType
+ certifiedICAO: CodeYesNoType
+ controlType: CodeMilitaryOperationsType
+ dateMagneticVariation: DateYearType
+ designator: CodeAirportHeliportDesignatorType
+ designatorIATA: CodeIATAType
+ fieldElevation: ValDistanceVerticalType
+ fieldElevationAccuracy: ValDistanceVerticalType
+ landingDirectionIndicator: CodeYesNoType
+ locationIndicatorICAO: CodeICAOType
+ lowestTemperature: ValTemperatureType
+ magneticVariation: ValMagneticVariationType
+ magneticVariationAccuracy: ValAngleType
+ magneticVariationChange: ValMagneticVariationChangeType
+ name: TextNameType
+ privateUse: CodeYesNoType
+ referenceTemperature: ValTemperatureType
+ secondaryPowerSupply: CodeYesNoType
+ transitionAltitude: ValDistanceVerticalType
+ transitionLevel: ValFLType
+ type: CodeAirportHeliportType
+ verticalDatum: CodeVerticalDatumType
+ windDirectionIndicator: CodeYesNoType
GroundLightSystem
«feature»
Lighting::TaxiwayLightSystem
+ position: CodeTaxiwaySectionType
+associatedAirportHeliport
0..1
isSituatedAt
0..*
isProvidedFor
+lightedTaxiway
0..1
0..*
«feature»
Taxiway
+ abandoned: CodeYesNoType
+ designator: TextDesignatorType
+ length: ValDistanceType
+ type: CodeTaxiwayType
+ width: ValDistanceType
+ widthShoulder: ValDistanceType
+markedTaxiway
0..1
+associatedTaxiway
0..1
+taxiwayLocation
0..1
isMarkedOn
hasSurfaceDescribedBy
isLocatedOn
0..*
Marking
«feature»
Markings::TaxiwayMarking
+ markingLocation: CodeTaxiwaySectionType
+surfaceProperties
0..1
describesPartOf
«object»
Airport/Heliport::SurfaceCharacteristics
+ classLCN: ValLCNType
+ classPCN: ValPCNType
+ composition: CodeSurfaceCompositionType
+ evaluationMethodPCN: CodePCNMethodType
+ maxTyrePressurePCN: CodePCNTyrePressureType
+ pavementSubgradePCN: CodePCNSubgradeType
+ pavementTypePCN: CodePCNPavementType
+ preparation: CodeSurfacePreparationType
+ surfaceCondition: CodeSurfaceConditionType
+ tyrePressureSIWL: ValPressureType
+ weightAUW: ValWeightType
+ weightSIWL: ValWeightType
0..*
«feature»
Apron::
DeicingArea
hasSurfaceDescribedBy
0..1
0..*
+surfaceProperties
0..*
isPaintedOn
+surfaceProperties
0..1
hasSurfaceDescribedBy
+markedElement
0..1
0..*
«feature»
TaxiwayElement
+ gradeSeparation: CodeGradeSeparationType
+ length: ValDistanceType
+ type: CodeTaxiwayElementType
+ width: ValDistanceType
0..*
0..*
hasExtent
hasExtent
+extent
0..1
+extent
0..1
Surface
«object»
Geometry::ElevatedSurface
+ elevation: ValDistanceVerticalType
+ geoidUndulation: ValDistanceSignedType
+ verticalAccuracy: ValDistanceType
+ verticalDatum: CodeVerticalDatumType

Figure 4. Taxiway Properties and Associations

In order to determine that distance, the spatial relation of all adjacent taxiways has to be analyzed. As seen in Figure 4, in AIXM, `Taxiway` features consist of a number of `TaxiwayElement` features which in turn contain a geometry (a surface). Taxiways do have properties for width and length, but no property for their spatial distance to the end (or start) of an adjacent runway. Hence, that value has to be calculated by a spatial distance operation taking two geometry objects as operands (see Figure 2 and Figure 4):

- the start of runway; available through `Runway←uses.RunwayDirection.startingElement.RunwayElement.extent.ElevatedSurface`
- the taxiway geometry; available through `Taxiway←describesPartOf.TaxiwayElement.extent.ElevatedSurface`

Please note that ← denotes a reverse association, e.g. that `TaxiwayElement` contains an association to `Taxiway`.

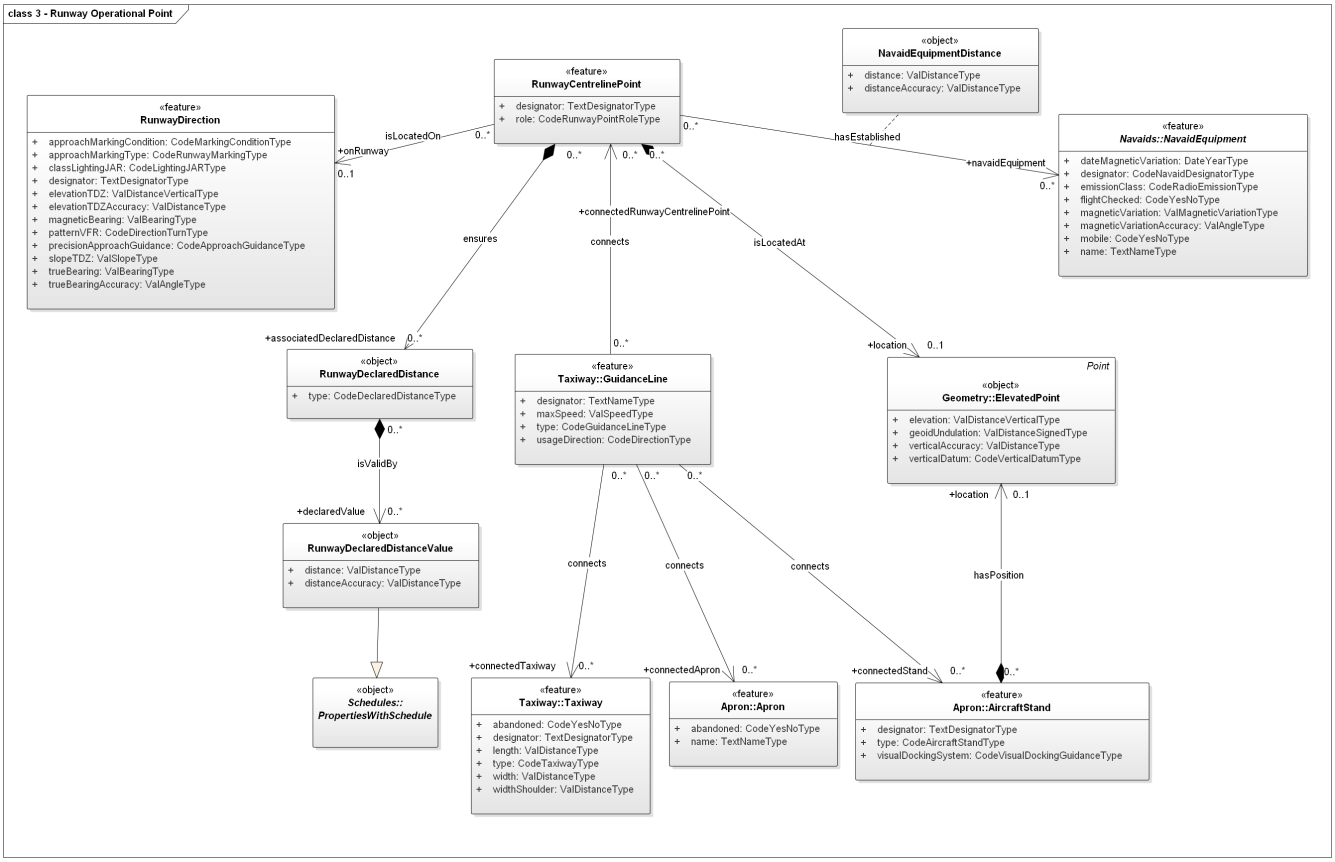

class 3 - Runway Operational Point
«feature»
RunwayDirection
+ approachMarkingCondition: CodeMarkingConditionType
+ approachMarkingType: CodeRunwayMarkingType
+ classLightingJAR: CodeLightingJARType
+ designator: TextDesignatorType
+ elevationTDZ: ValDistanceVerticalType
+ elevationTDZAccuracy: ValDistanceType
+ magneticBearing: ValBearingType
+ patternVFR: CodeDirectionTurnType
+ precisionApproachGuidance: CodeApproachGuidanceType
+ slopeTDZ: ValSlopeType
+ trueBearing: ValBearingType
+ trueBearingAccuracy: ValAngleType
«feature»
RunwayCentrelinePoint
+ designator: TextDesignatorType
+ role: CodeRunwayPointRoleType
«object»
NavaidEquipmentDistance
+ distance: ValDistanceType
+ distanceAccuracy: ValDistanceType
«feature»
Navaids::NavaidEquipment
+ dateMagneticVariation: DateYearType
+ designator: CodeNavaidDesignatorType
+ emissionClass: CodeRadioEmissionType
+ flightChecked: CodeYesNoType
+ magneticVariation: ValMagneticVariationType
+ magneticVariationAccuracy: ValAngleType
+ mobile: CodeYesNoType
+ name: TextNameType
isLocatedOn
+onRunway
0..1
0..*
hasEstablished
+navaidEquipment
0..*
ensures
+connectedRunwayCentrelinePoint
connects
isLocatedAt
+associatedDeclaredDistance
0..*
+location
0..1
«object»
RunwayDeclaredDistance
+ type: CodeDeclaredDistanceType
«feature»
Taxiway::GuidanceLine
+ designator: TextNameType
+ maxSpeed: ValSpeedType
+ type: CodeGuidanceLineType
+ usageDirection: CodeDirectionType
Point
«object»
Geometry::ElevatedPoint
+ elevation: ValDistanceVerticalType
+ geoidUndulation: ValDistanceSignedType
+ verticalAccuracy: ValDistanceType
+ verticalDatum: CodeVerticalDatumType
isValidBy
+declaredValue
«object»
RunwayDeclaredDistanceValue
+ distance: ValDistanceType
+ distanceAccuracy: ValDistanceType
connects
connects
connects
hasPosition
+location
0..1
«object»
Schedules::
PropertiesWithSchedule
+connectedTaxiway
+connectedApron
+connectedStand
«feature»
Taxiway::Taxiway
+ abandoned: CodeYesNoType
+ designator: TextDesignatorType
+ length: ValDistanceType
+ type: CodeTaxiwayType
+ width: ValDistanceType
+ widthShoulder: ValDistanceType
«feature»
Apron::Apron
+ abandoned: CodeYesNoType
+ name: TextNameType
«feature»
Apron::AircraftStand
+ designator: TextDesignatorType
+ type: CodeAircraftStandType
+ visualDockingSystem: CodeVisualDockingGuidanceType

Figure 5. Runway Operational Associations

The next step is to find a link between taxiways and runways. There are two options:

- model navigation: find an association path in the model
- spatial calculation: find all taxiways that spatially touch the runway

As Figure 5 shows, an association path from `Taxiway` to `Runway` is given via the objects `RunwayDirection`, `RunwayCentrelinePoint` and `GuidanceLine`.

For the spatial calculation, the extents of all `RunwayElement` objects linked through association `describesPartOf` to `Runway` have to be considered.

We choose the spatial calculation, as this the topic of the section. This gives us:

```
It is prohibited that an AirportHeliport [...](1) has not at-least-one Runway that
isSituatedAt it with (at-least-one RunwayElement that describesPartOf it and
extent.ElevatedSurface touches[Taxiway...extent.ElevatedSurface](2)) and
RunwayDirection that uses it with startingElement.RunwayElement.extent.ElevatedSurface
is-within-distance of X to[Taxiway...extent.ElevatedSurface](3)
```

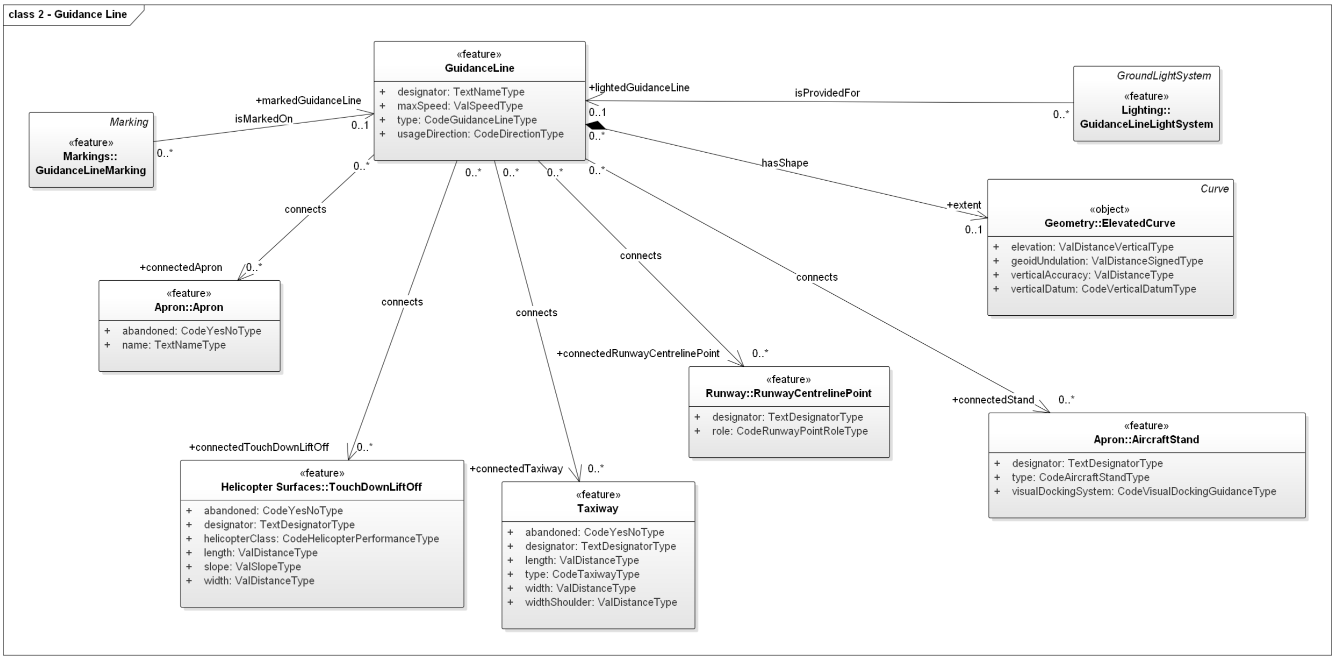
Where the parts in square brackets stand for

- (1): the constraints on availability and usage as identified in Landing Distances and Runways
- (2): the geometry of any Taxiway (at-least-one Taxiway …)
- (3): the geometry of the **same** Taxiway as in (2)

It isn't possible to further refine the constraint with the present vocabulary and grammar. To refine (2), we need a possibility to combine an quantification and a navigation. For (3), we need an identifier for reference.

7.1.3. Limitations in turning radius of the aircraft

The problems of navigating through the complex model with SBVR and formulating complex constraints between different elements of it have been sufficiently covered by the use cases above. Here, we focus on the spatial constraint instead.



class 2 - Guidance Line
«feature»
GuidanceLine
+ designator: TextNameType
+ maxSpeed: ValSpeedType
+ type: CodeGuidanceLineType
+ usageDirection: CodeDirectionType
Marking
«feature»
Markings::
GuidanceLineMarking
+markedGuidanceLine
isMarkedOn
0..1
0..*
+lightedGuidanceLine
isProvidedFor
0..1
0..*
GroundLightSystem
«feature»
Lighting::
GuidanceLineLightSystem
hasShape
0..*
+extent
0..1
Curve
«object»
Geometry::ElevatedCurve
+ elevation: ValDistanceVerticalType
+ geoidUndulation: ValDistanceSignedType
+ verticalAccuracy: ValDistanceType
+ verticalDatum: CodeVerticalDatumType
connects
+connectedApron
0..*
0..*
«feature»
Apron::Apron
+ abandoned: CodeYesNoType
+ name: TextNameType
connects
+connectedTouchDownLiftOff
0..*
0..*
«feature»
Helicopter Surfaces::TouchDownLiftOff
+ abandoned: CodeYesNoType
+ designator: TextDesignatorType
+ helicopterClass: CodeHelicopterPerformanceType
+ length: ValDistanceType
+ slope: ValSlopeType
+ width: ValDistanceType
connects
+connectedTaxiway
0..*
0..*
«feature»
Taxiway
+ abandoned: CodeYesNoType
+ designator: TextDesignatorType
+ length: ValDistanceType
+ type: CodeTaxiwayType
+ width: ValDistanceType
+ widthShoulder: ValDistanceType
connects
+connectedRunwayCentrelinePoint
0..*
0..*
«feature»
Runway::RunwayCentrelinePoint
+ designator: TextDesignatorType
+ role: CodeRunwayPointRoleType
connects
+connectedStand
0..*
0..*
«feature»
Apron::AircraftStand
+ designator: TextDesignatorType
+ type: CodeAircraftStandType
+ visualDockingSystem: CodeVisualDockingGuidanceType

Figure 6. AIXM 5.1.1 GuidanceLine UML Class Diagram

The turning radius of an aircraft might be too big for the aircraft to operate on a taxiway describing a curve. The validation needs to check whether the minimum radius of some part of a taxiway is bigger than an acceptable value. Figure 6 shows that `Taxiway` objects are connected to `GuidanceLine` objects which contain a geometry object of type `ElevatedCurve`. A curve geometry in GML is made up of segments, where each segment can either be

- a line described by a set of points
- different types of arcs, some of them containing a radius

If all segments would be types of arcs with a radius property such as `ArcByCentrePoint`, the constraint will be

```
each extent.ElevatedCurve.segments.ArcByCenterPoint.radius is-greater-than X
```

However, in the general case a combination of straight and curved elements are likely, where the connection points of two segments constitute a singularity regarding the required turning radius. To get a realistic constraint, some sort of smoothing of the guidance line has to be applied in advance, and the minimum radius derived from it. This complex operation cannot be described with existing spatial operators. A new specialized spatial operator has to be introduced.

7.2. Use Case 2: Identification of airports along the trajectory of a flight

This use case is relevant for two scenarios: flight plan validation and flight plan creation. The first is a regular validation scenario, except that information from different sources have to be combined: flight plan data and static aeronautical data. When *creating* a flight plan, on the other hand, similar rules apply, but are used to identify suitable objects instead of validating them.

7.2.1. Flight plan validation

Identification of airports along the flight trajectory is necessary to comply with Extended Operations (ETOPS) rules for emergency landings.

Trajectories of concrete flights are not found in AIXM, which defines static data, but in FIXM. In order to provide a stable validation rule, it is important that flight data from FIXM can be unambiguously linked to corresponding AIXM data. However, this is not provided in the current version of FIXM, V3.0.1, in which for example airports are referenced by their name only. For a stable resolvable link to AIXM, their unique identifiers (gml:identifier UUIDs) have to be provided. This is one of the topics discussed for the next version of FIXM in this testbed (see [9]).

For this use case, we therefore assume that the trajectory is given as a list of geo-locations starting at the departure airport and ending at the destination airport (or alternate).

The objective: check if suitable emergency airports are in acceptable reach at every time of the flight, taking into applicable ETOPS rules. The ETOPS rules define a maximum distance to the next suitable airport suitable for emergency landing, depending on the type of the aircraft. The determination of that distance from the aircraft type is not considered here, as we want to focus on the geo-spatial constraints that are involved. An example is shown in Figure 7.

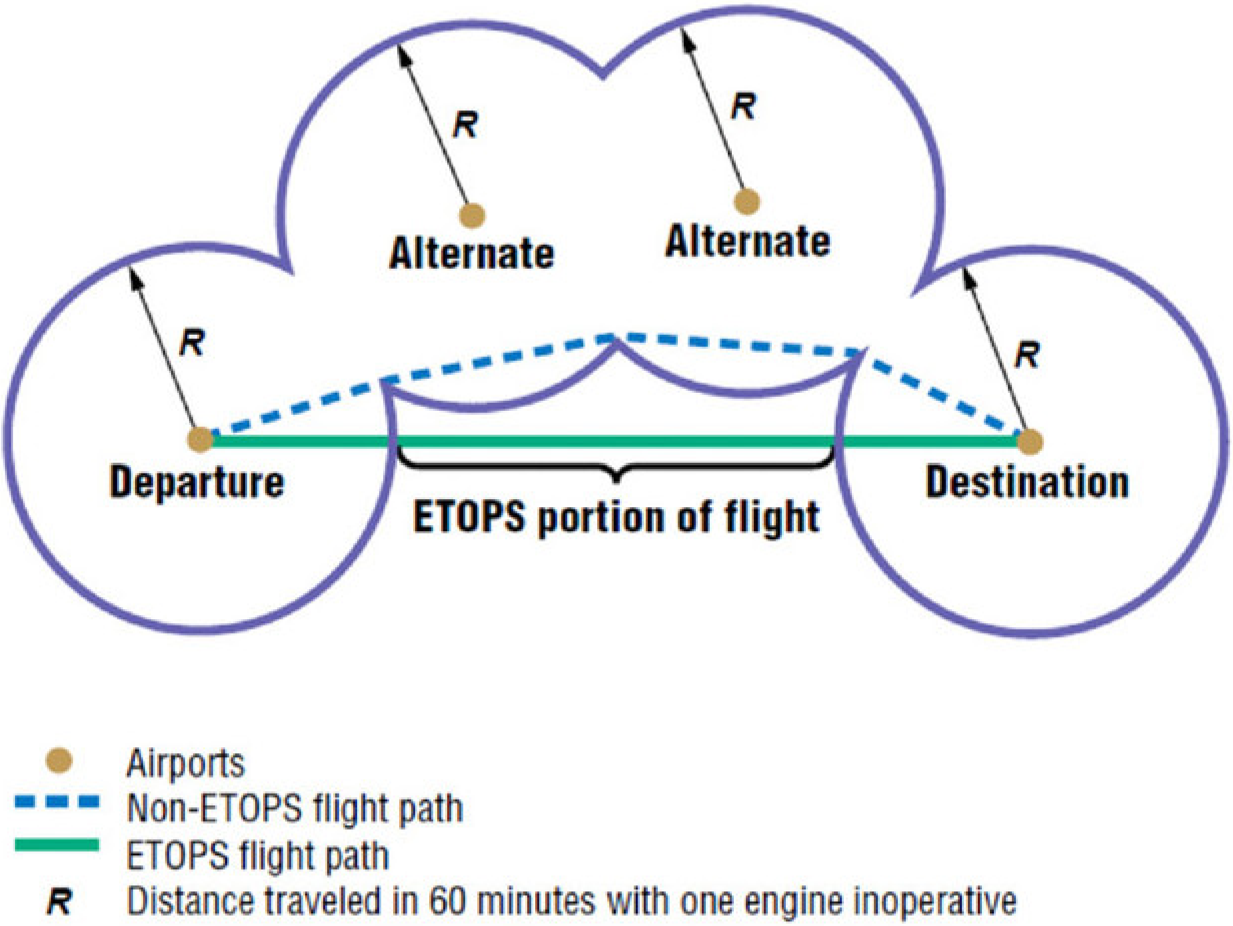


Figure 7. Example for the validation of flight route according to ETOPS regulations

A pseudo algorithm for this could be formulated as follows:

1. For each suitable airport in ETOPS range of the flight route, construct a circle with a radius equal to the maximum ETOPS range
2. Construct the union of all these circles
3. Check if the flight route is fully contained in that union

SBVR rules have to be formulated in one sentence. We did not find a way to accomplish that. A first draft could look like this:

```
It is prohibited that a "flight route" has at-least-one "route part" disjoint "the
union of all circles with the radius of ETOPS range around all suitable airports in
ETOPS range of the flight route"
```

The complexity lies in the last part, where a geometry has to be constructed as a union of several other geometries fulfilling certain constraints. Unfortunately, SBVR does not support any geometry construction yet, and neither depending on its constituents on elements in the rule (here "the flight route", i.e. a list of geo-locations connecting the departure with the destination airport). Even if that might be possible in a future version of SBVR for AIXM, the restriction that a rule is formulated in a single sentence impedes the readability. Instead, for rules with such a complexity, the breakdown in smaller "steps", as in the pseudo algorithm example, is preferred.

7.2.2. Flight plan creation

The creation of a flight plan involves the identification of suitable features. This is different from a validation scenario, where existing data is checked for consistency. These differences, and how they can be supported by SBVR, is covered in Filtering Data Using SBVR. Regarding the actual SBVR expressions involved, refer to Flight plan validation above.

7.3. Use Case 3: Validation of an airport’s address against its geo-location

This use case was identified in [1] and is an example for a rather simple application of the `within` operator. A concrete validation rule could look like this:

```
It is obligatory that each AirportHeliport with contact.address.country equal-to
'GERMANY' shall have ARP within 'POLYGON((5.75 55.5,15 55.5,15 47,5.75 47,5.75
55.5))'.
```

Here, it is checked if the ARP of an airport with an address in Germany lies in the German territory, provided as geometry object. Such a rule could be formulated for every country in the world, just by varying the geometry part, and hence giving full coverage.

The geometry is given in WKT [10] format, which is a well-supported standard for representing geometries in a textual format.

This use case was taken for proof of concept if existing tools in the area of SBVR and AIXM can be extended to support spatial operators and what are the challenges in doing so. The following tools were involved:

- ShapeChange (http://shapechange.net): an open source tool that converts application schemas in UML to GML application schemas, Schematron schemas, JSON schemas and many other representations
- m-click.aero’s AIXM Validator Service (https://m-click.aero/products/aeronautical-data-validation-platform/): a web service for generating comprehensive validity reports for XML-encoded AIXM datasets

7.3.1. Extending ShapeChange to support spatial operators

The SBVR implementation of ShapeChange follows a three step process:

1. Parse the SBVR rule with an ANTLR (http://www.antlr.org) grammar
2. Generate a first-order logic (FOL) representation of the rule
3. Convert the FOL representation into Schematron

Changes and extensions had to be applied on each of the steps:

- the grammar was extended to support the spatial operators as new predicates, and WKT geometries as literals

- the FOL language was extended to support binary function predicates
- the FOL-to-Schematron converter was extended to support custom binary functions in prefix notation

The result is Schematron code enriched with custom XPath functions, which were implemented by m-click.aero’s AIXM Validator Service.

7.3.2. Extending m-click.aero’s AIXM Validator Service

m-click.aero’s AIXM Validator Service uses a custom Schematron implementation to allow for more flexibility than common XSLT-based implementations. The main work was done in the embedded XPath 2.0 engine, which had to be extended by custom XPath functions for spatial operations. Multiple types of operands had to be supported:

- GML geometries, including the AIXM extended versions with elevation
- WKT literals

The following spatial operators were implemented:

- disjoint
- touches
- intersects
- crosses
- within
- contains
- overlaps
- covers
- covered-by

7.3.3. Summary

The proof of concept was successfully made: the implementation path set by the toolset ShapeChange + m-click.aero’s AIXM Validator Service is extendable to support spatial operators. Despite the complexity of the task, where several intermediate levels and representations were affected, the extension towards the application of custom XPath functions is promising for future enhancements, as they give a lot of power and flexibility.

8. Filtering Data Using SBVR

In the RFQ, it was stated that rules should be used to *identify* airports along a trajectory that fulfill certain constraints. While it may seem natural to use business rules for identification of entities out of a collection, SBVR rules are not designed to search for data. Applying SBVR rules means checking existing data for consistency. If one or more rules are broken, this is reported as an error. Still, this use case could also benefit from the advantages of SBVR as a machine-readable language for non-technically focused domain experts.

Two approaches are conceivable. A minimalist version, where it is assumed that no insight in the SBVR evaluation is given. This approach should always work, but might be inefficient and unintuitive.

Therefore, another optimized concept was evaluated, where the filtering-relevant parts of the SBVR evaluation algorithm were identified.

8.1. Minimalist approach

The validation error report usually contains the inconsistent elements together with the rule(s) violated. In order to use SBVR rules for filtering data, we have to change the meaning of "breaking a rule" to "not matching the filter". That means the result of the filter operation is the input set of data subtracted by all elements contained in the error report (i.e. elements that violate any rule). The basic algorithm is outlined in Figure 8.

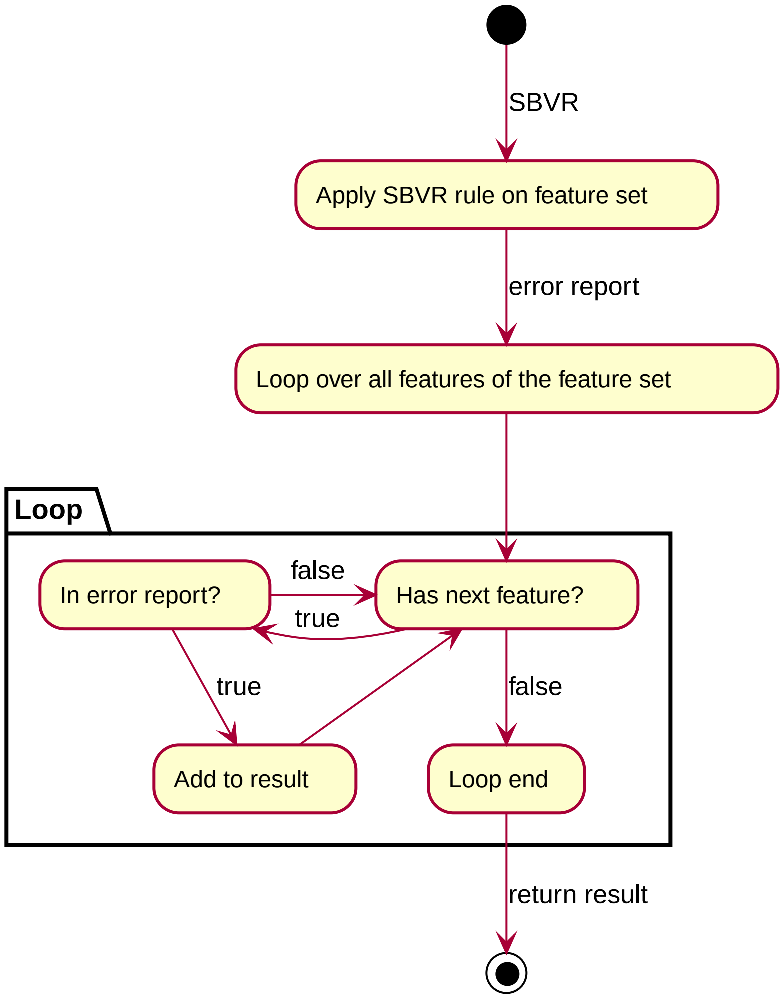


Figure 8. A minimalist process for filtering data with SBVR

8.1.1. Discussion

With this approach, any SBVR rule might be used as a filter, and it should work with any SBVR processor. However, it is inefficient, because the complete set of features (perhaps initially filtered by type) is processed twice.

8.2. Optimized approach

Most of the SBVR rules found in [5] follow one of the following schemes:

1. `It is prohibited that each <feature type> … <conditions and constraint>`
2. `It is obligatory that each <feature type> … <conditions and constraint>`
3. `Each <feature type> … <conditions and constraint>`

These schemes can be easily transformed into each other: the first scheme is the negation of the second and vice versa, and the last scheme differs from the second only in the wording. Further, the `<feature type>` part is not different from any other condition, as it could be reformulated as follows

```
It is prohibited/obligatory that feature with type <feature type> …
```

Hence, all rules could be reformulated to the scheme

`It is obligatory that each feature … <conditions and constraint>`

In that form, only conditions and constraints are left. *Conditions* select the features for which the *constraints* shall be fulfilled. That means, at some point in the evaluation of an SBVR rule a selection of features has to take place, which is nothing different from a filtering process. An example:

```
It is obligatory that each AirportHeliport with contact.address.country equal-to
'GERMANY' shall have ARP within 'POLYGON((5.75 55.5,15 55.5,15 47,5.75 47,5.75
55.5))'.
```

Here, the selection is defined by the term `each AirportHeliport with contact.address.country equal-to 'GERMANY'`.

8.2.1. Discussion

All SBVR rules contain a filter expression which has to be evaluated by the SBVR processor. If only the filter expression is used, and if the sub-module of an SBVR processor interpreting that expression could be isolated, an efficient condition-based identification of features in a human-friendly language is possible.

In the context of Web Feature Services [7], such "SBVR filter expressions" could be used as an alternative to filters based on the Filter Encoding Implementation Specification [8].

9. Dealing with Temporality in SBVR

Temporality in AIXM is based on the AIXM Temporality Model, which manifests itself in two areas:

1. the "time slice" objects of AIXM
2. recurring schedules, represented by objects of type `PropertiesWithSchedule`

Please note that the time slice objects in XML have no counterpart in AIXM UML Model. [1] states that this is problematic for SBVR, because it is designed to operate on the model, not an implementation schema. It further suggest two solutions for that deficiency (see [1], 7.2):

- The rule itself contains the selection of time slices, as required.
- Metadata for a rule states which time slice interpretations are to be checked

While this may be adequate for handling the time slice objects AIXM, it is not suitable for dealing with recurring schedules.

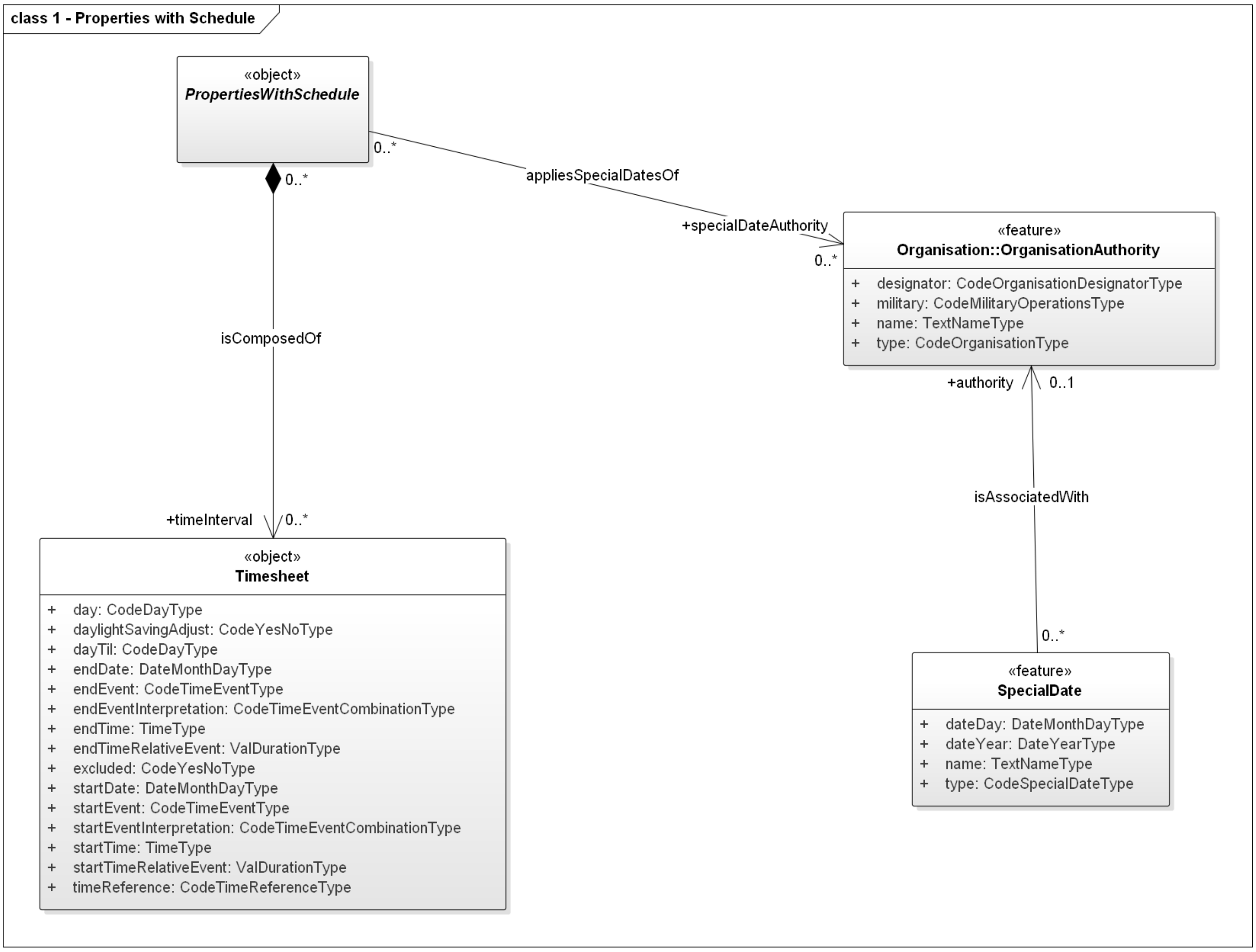

class 1 - Properties with Schedule
«object»
PropertiesWithSchedule
0..*
appliesSpecialDatesOf
+specialDateAuthority
0..*
«feature»
Organisation::OrganisationAuthority
+ designator: CodeOrganisationDesignatorType
+ military: CodeMilitaryOperationsType
+ name: TextNameType
+ type: CodeOrganisationType
0..*
isComposedOf
+timeInterval
0..*
+authority
0..1
isAssociatedWith
0..*
«object»
Timesheet
+ day: CodeDayType
+ daylightSavingAdjust: CodeYesNoType
+ dayTil: CodeDayType
+ endDate: DateMonthDayType
+ endEvent: CodeTimeEventType
+ endEventInterpretation: CodeTimeEventCombinationType
+ endTime: TimeType
+ endTimeRelativeEvent: ValDurationType
+ excluded: CodeYesNoType
+ startDate: DateMonthDayType
+ startEvent: CodeTimeEventType
+ startEventInterpretation: CodeTimeEventCombinationType
+ startTime: TimeType
+ startTimeRelativeEvent: ValDurationType
+ timeReference: CodeTimeReferenceType
«feature»
SpecialDate
+ dateDay: DateMonthDayType
+ dateYear: DateYearType
+ name: TextNameType
+ type: CodeSpecialDateType

Figure 9. AIXM 5.1.1 Properties with Schedule Class Diagram

Figure 9 shows the properties of type `PropertiesWithSchedule`. It allows one to define recurring schedules through an unlimited number of `Timesheet` objects, where each can be based on

- dates (day + month, weekdays, working days, holidays)

- times (point in time and time intervals)
- events (sunrise, sunset, other)
- other `Timesheet` objects with exclusions

The list is not intended to be complete, but to show the complexity. If an SBVR rule is based on the current state of a feature or object, according to [1] it has to constrain time slice types, valid times and possibly recurring schedules. For simplification, we suggest a different approach. Using the concept of SNAPSHOT time slices for time periods from [6], we are able to move the calculation of the current state outside the SBVR rules. For this to work, each rule that should be valid for the whole lifetime of a feature (or lifetimes of multiple features) shall be applied to the complete set of SNAPSHOT time slices covering the lifetime of the feature (or lifetimes of multiple features). If recurring schedules are involved, they can be resolved by applying the same algorithm as in [6], section 8.4 ("evaluateSchedules").

Appendix A: Revision History

Table 2. Revision History

Date	Release	Editor	Primary clauses modified	Descriptions
February 16, 2016	.1	T. Thomas	all	initial version
April 29, 2016	.2	T. Thomas	all	draft
May 2, 2016	.3	T. Thomas	8, 9	more detail, added diagrams
July 7, 2016	.4	T. Thomas	all	release candidate
October 12, 2016	.5	T. Thomas	8, references, bibliography	minor corrections, reference styling

Appendix B: Bibliography

[1] OGC 15-024r2, OGC® Testbed 11 Aviation - Guidance on Using Semantics of Business Vocabulary and Rules (SBVR) Engineering Report (2015)

[2] SESAR Joint Undertaking: Guidance on Writing AIRM Constraints, Edition 01.00.01 (2014)

[3] OMG: Semantics of Business Vocabulary and Rules (SBVR) 1.1 (2013)

[4] Slides from ATIEC: Digital NOTAM Eurocontrol perspective, Porosnicu, E., http://aixm.aero/sites/aixm.aero/files/imce/library/ATIEC_2015/19_day2_digital_notam_eurocontrol_perspective.pdf (2015)

[5] AIXM 5.1 - Business Rules (data verification) - Using SBVR and Schematron V0.5 (2015)

[6] OGC 12-027r2, WFS Temporality Extension (2013)

[7] OGC 09-025r2, Web Feature Service 2.0 Implementation Standard (2014)

[8] OGC 04-095, Filter Encoding Implementation Specification (2005)

[9] OGC 16-028, FIXM GML Engineering Report (2016)

[10] OGC 06-103r4, OpenGIS ® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture (2011)