Change ID: 5.1-41

Summary

The Airport/Heliport, Airspace, Route, Procedure and AerialRefuelling features have associated "usage" classes, which model restrictions or guidance related to their operations. This change proposal is meant to consolidate, simplify and harmonise these various "usage" models.

Background

In AIXM 5.0 the term "usage" defines restrictions or rules that affect the operations of a feature such as an Airport , a Route , an Airspace a Procedure or an AerialRefuelling . In general, this permits, forbids or restricts the operations based on conditions such as:

- Flight type - military, civilian, commercial, general aviation;
- Aircraft type - jet, helicopter, piston, engines;
- Operation - departures, arrivals;
- MET conditions;
- Time, etc.

In addition, AIXM 5 includes a concept called Flight Restrictions , which models rules meant to regulate the use of the route network, by identifying a set of flights and either forbidding them on a particular routing or obliging them to follow one routing out of a set of mandatory alternatives.

Rationale for the change

There are several issues that require an improvement of the current model. These are discussed in more detail in the following sections and include:

- some of the "usage" classes model in fact the activation status of the feature and not really usage restrictions. We define the "activation" as the availability of a feature for operational use. The "usage" is an indication of the subset of flight operations that can use the specified aeronautical feature. The current model does not make the necessary distinction between the two;
- the disharmonious approaches between the different "usage" classes in the current model;
- the feed-back from the Digital NOTAM Trials that identified difficulties and missing capabilities.

"Usage" versus "activation status" for AirportHeliport

The AirportHeliport Usage class contains a set of restrictions and permissions for the usage of an associated AirportHeliport. The same restriction may apply at once to several movement area elements (e.g., a specific runway, taxiway, taxiway element, aircraft stand, etc.), including the airport itself.

With the current model, a closed runway would be encoded as an **AirportHeliportUsage** restriction associated with the RunwayDirection. In most situations, a new Baseline would need to be created, valid for the time of the closure only. But the runway might already have other Baseline usage restrictions, for example based on aircraft type, meteo conditions, time, which apply to other features and also to that runway. This creates a complex hierarchy of dynamic Baseline restrictions, which is hard to control and interpret.

In parallel, a number of airport elements also have a "status" attribute, which partially overlaps with the AirportHeliportUsage concept. This is the case for Apron , TouchDownLiftOff , ArrestingGear , DeicingArea , Road , RunwayBlastPad , etc.

Based on the experience from the Digital NOTAM Trials, it seems to be more appropriate to model the active/ inactive status of a runway (or other airport element) as a property of the feature. Only the usage restrictions based on aircraft weight, type, wingspan, etc. should be modelled as AirportHeliportUsage.

"Activation status" instead of Airspace "usage"

The Airspace Usage does not really model usage restrictions. It models the activation status (level and time plus type of activity) of the area. The "type" attribute of the Airspace class already gives a strong indication about the kind of operations permitted in the airspace. For example, a P (Prohibited) area can not be used for flying.

There exists only one attribute ("trafficAllowed") that indicates a kind of usage restriction, when the airspace is active. This single attribute was found insufficient in order to model more complex flying restrictions that can affect an active airspace. On the other side, the usage restrictions of an active airspace can be described using the FlightRestriction class. There is no real need for a separate "trafficAllowed" attribute in the AirspaceUsage feature.

Therefore, it is proposed to change the model so that the active/inactive status of an airspace (including levels, type of activity) becomes a property of the Airspace feature. More complex usage restrictions of an active Airspace are covered by the FlightRestrictions model.

"Availability" instead of Route portion "usage"

The RoutePortionUsage models the "availability" of the route and much less flying restrictions based on aircraft type of flight type. Such flying restrictions, where they exist, are already covered through the FlightRestriction class, which is also part of the Routes sub-package. Currently, the difference between Route Usage and Flight Restriction is that the Route Usage does not depend on the origin/destination of the flight, while the Flight Restriction does. But this is a very small difference, which does not justify the existence of two different approaches.

It would be clearer if the RoutePortionUsage was replaced with a complex "availability" property of the RouteSegment feature, while all the route usage restrictions that depend on flight or aircraft type were covered through the FlightRestrictions class.

The RoutePortionUsage also includes the minimum and maximum permissible flight levels, navigation type (RNAV, Conventional) and RNP values. These properties are the result of the route design and can be specific for each route segment, for example, when necessary to avoid certain areas or obstructions. This proposal recommends that these properties be relocated to the RouteSegment feature.

"Availability" instead of Procedure "usage"

Similar to the RoutePortionUsage, the ProcedureUsage describes operational availability of the associated Procedure.

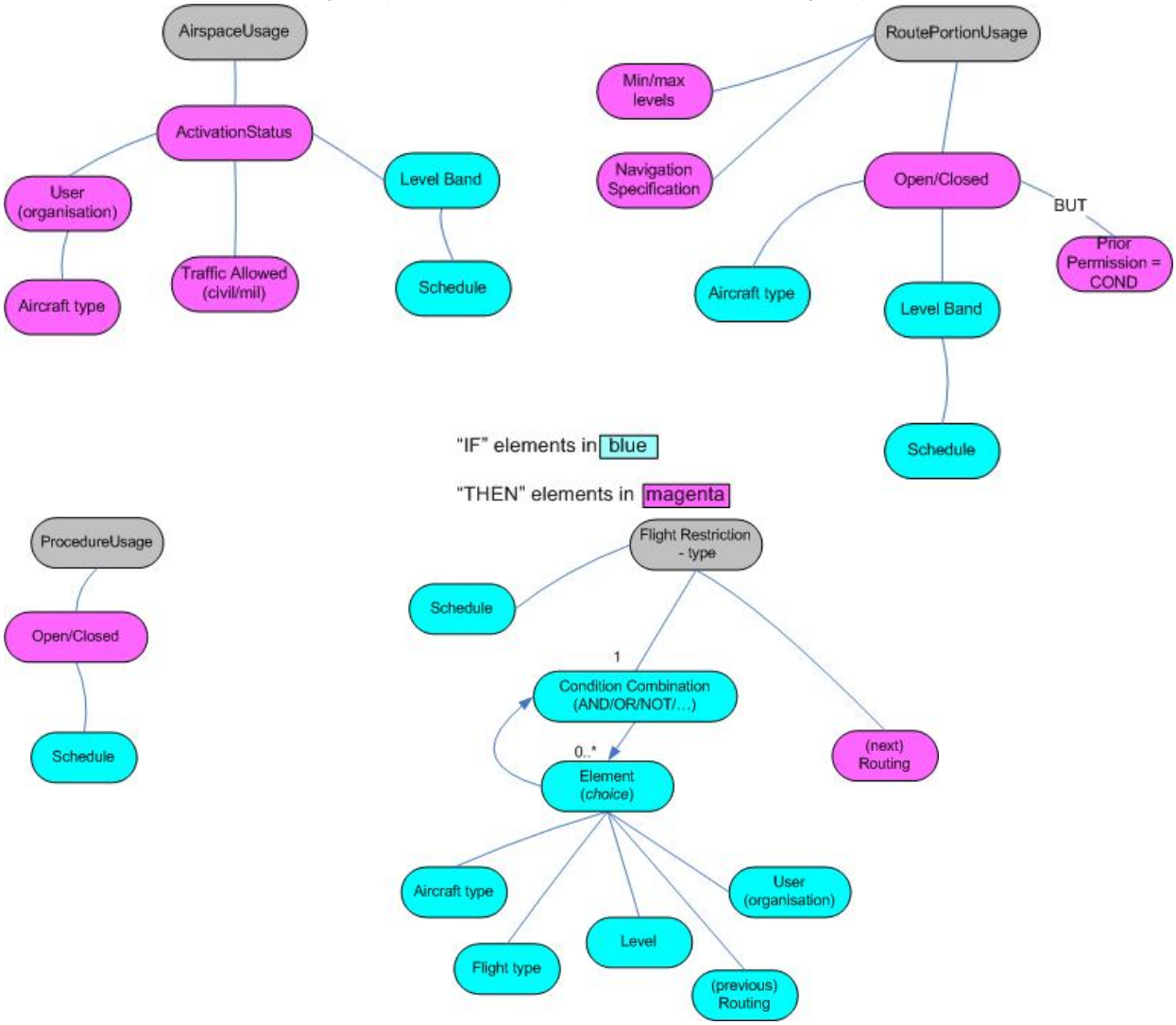
It is therefore proposed to be renamed ProcedureAvailability and to follow the same design pattern as for RouteSegmentAvailability. Eventual flying restrictions associated with that procedure are already covered through the FlightRestriction class.

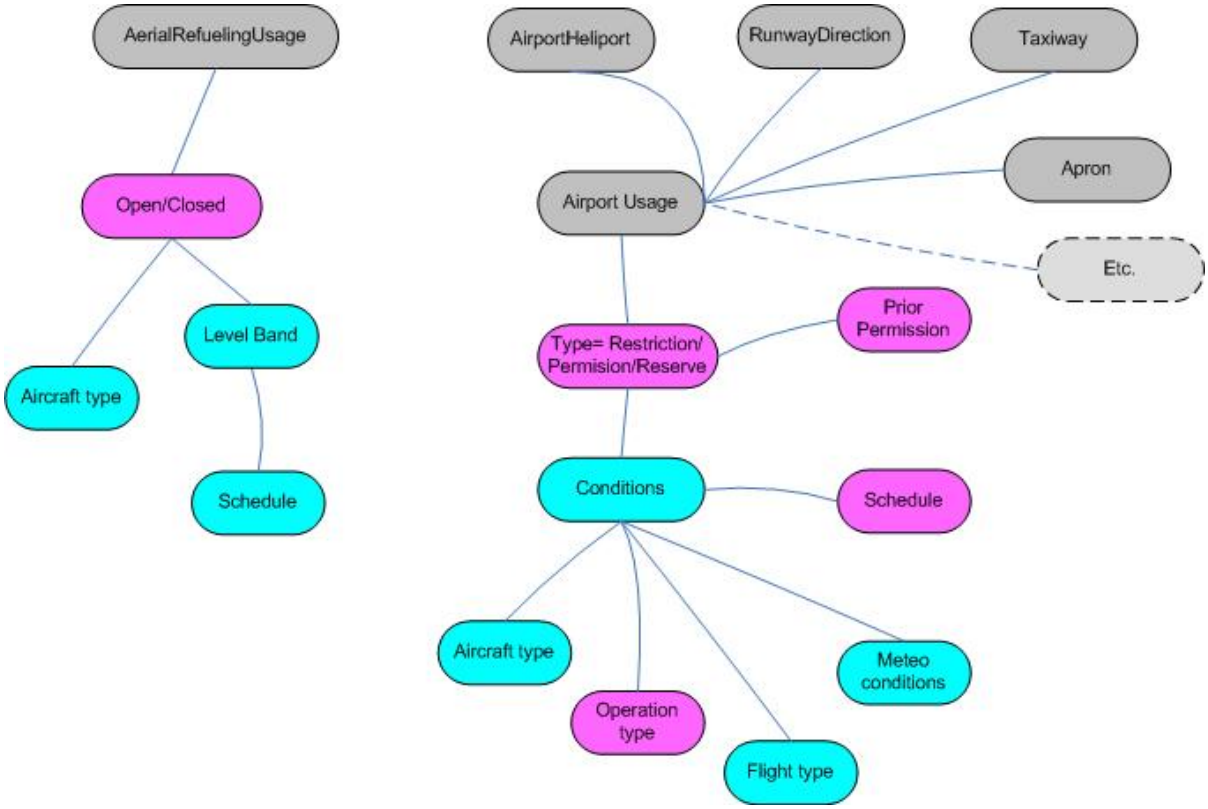
"Availability" versus "usage restrictions" for Aerial Refuelling

The AerialRefuellingUsage class describes the availability of an established AerialRefuelling procedure, in particular the levels, the times and the aircraft types. This is quite similar to the route availability model and the two could be covered by a single "RouteAvailability" class. Additional usage restrictions, such as based on aircraft type, should be modelled as FlightRestrictions. For this purpose, the condition and routings of the Flight Restrictions model shall include AerialRefuelling.

Disharmonious "usage" models

The diagrams below present an abstraction of the current usage models, using just high level components such as "flight type", "aircraft type", "METEO conditions", "schedule", etc. These components play either an "if" role (as those coloured in blue in the diagrams) or a "then" role (as those coloured in magenta).





The current models do not have a logical separation into a "conditions selection branch" and a "restriction branch". Although there are many common building blocks, each usage model arranges them differently. This makes the usage models quite hard to understand.

The model would be much clearer if the usage models made a clear separation between the "if" (criteria that are used to indicate which flights are affected) and "then" (restrictions that apply to the corresponding flights). Also, where common concepts are used, such as meteo conditions, flight type, aircraft type, etc., they should be used in a similar way.

Change proposal principles

It is proposed that the model is re-organised, in order to make a clear distinction between "activation" and "usage":

1. Activation - an indication of the operational status of the aeronautical feature. The availability of a feature for operational use.
2. Usage - An indication of the subset of flight operations that can use the specified aeronautical feature, when that feature is active. A subset of flights is allowed or restricted operational access to an aeronautical feature based on a set of conditions and restrictions.

In other words the activation status is a property of the aeronautical feature. The "usage" are criteria or limitations that a flight must fulfil or achieve in order to access the feature.

This has the following practical consequences:

- "activation" or "availability" to be modelled as a property of the feature concerned (AirportHeliport, RunwayDirection, RouteSegment, Airspace, etc.). This might need to be a complex property, because of the need to include levels, schedules and eventually type of operation;
- airspace, procedures, aerial refuelling and route usage rules to be covered through the existing FlightRestrictions class, which might need to be upgraded with a few more capabilities, such as "prior permission", "MET conditions", etc.
- airport/heliport usage rules to be re-modelled following a similar "if..., then..." pattern as for the FlightRestriction class. This will facilitate both the understanding of the model and its implementation.

Other issues

Airport Operations

The list of values of the CodeAirportHeliportOperationType data type is applicable to the AirportHeliportUsage associated with any element, including Apron, Taxiway, AircratfStand. This includes LANDING, TAKEOFF, ALTN_LANDING, AIRSHOW, TOUCHGO, etc. But these operations apply only to manoeuvring area elements (runway, taxiway, TLOF, seaplane landing area). Therefore, the usage rules for non-manoeuvring area elements (apron, de-icing area, aircraft stand, etc.) does not require an 'operations' attribute.

OperationalStatus of AirportHeliport surfaces

The current list of values of the CodeStatusSurfaceType includes the values "ACTIVE"/"INACTIVE" but also "ABANDONED", "WORK_IN_PROGRESS" and "PARKED". This list of values is applicable to Apron, ArrestingGear, DeicingArea, Road, RunwayBlastPad, RunwayProtectArea, TaxiHoldingPosition, TouchDownLiftOff.

"ABANDONED" indicates a surface that is still physically present and visible, although probably in deteriorated condition and no longer used operationally. This approach is different from the one applied to the AirportHeliport class, which has an "abandoned" attribute, with values yes/no. The value "abandoned" makes sense only for physical surfaces, such as runway, taxiway, apron, road, TLOF. In addition, on certain circumstances, an abandoned feature can become temporarily usable. For example, an abandoned airport could be used for an Airshow or an abandoned taxiway could be used for parking. To enable such combinations, it is proposed to insert an "abandoned" attribute in all the classes that need it (Runway, Apron, Taxiway, TLOF, Road), similar to the one of the AirportHeliport class and to remove this value from the CodeStatusSurfaceType.

The values "WORK_IN_PROGRESS" and "PARKED" (aircraft) represent reasons for an operational limitation. They are already covered through the "reason" attribute of the AirportHeliportUsageLimitation class and shall be removed from this list of values.

In the new model, it is proposed that the operational status of an airport or one of its surfaces is designated using one of the following values:

- NORMAL = The facility operates with nominal parameters; it can be associated with several baseline usage limitations;
- LIMITED = The facility operates below its nominal parameters, with additional usage restrictions; typically, this is a temporary situation;
- CLOSED = The facility is not operational; it does not make sense to specify additional usage restrictions because they would be irrelevant.

Airspace activation status

The list of values of the statusActivation property seems to be mixing the workflow with the status of the Airspace. The legitimate values are:

- AVBL_FOR_ACTIVATION = The feature may be activated
- ACTIVE = The airspace is active (but it might not be in use yet)
- IN_USE = The airspace is actually used (inside the activation period)
- INACTIVE = The airspace is not active

The other values represent workflow steps and should be covered through either metadata or extensions, because they are very likely to be application or region specific. In addition, a new "intermittent" value is needed - indicating that the airspace is active, but not continuously.

Enhancements of the FlightRestriction model

In the context of the new usage models, the FlightRestriction class requires some small enhancements, in order to make it fully capable of representing usage limitations for Airspace and Routes:

- it is proposed to add Meteorology as another kind of elementary condition;
- it is proposed to add a priorPermission and ContactDetails for the Routing side, as some restricted areas may be penetrated with a prior permission;
- it is proposed to include AerialRefuelling into the choice of Conditions and Routings, in order to be able to express its usage restrictions.

Change proposal details

AirspaceActivation

In the "Airspace" package of the UML model, rename the diagram "4 - Airspace Activation" instead of "4 - Airspace Usage"

Delete the AirspaceUsage class.

Modify the class "AirspaceLayerUsage" as follows:

- rename as AirspaceActivation
- definition = " *Information about the operational status of the airspace*"
- insert new association: Airspace isActive [0..*] AirspaceActivation (role=" *activation*", aggregated by value) , navigable towards the AirspaceActivation class
- remove the attribute "trafficAllowed"
- rename the attribute "statusActivation" into "status"
- declare the AirspaceActivation class as a specialisation of the PropertiesWithSchedule

Modify the CodeAirspaceActivationType class:

- rename as CodeStatusAirspaceType
- changed definition = " *A coded list of values indicating the activation status of an Airspace*"
- remove the values REQUESTED and ALLOCATED
- add a new value INTERMITTENT = The Airspace is active but with periods of no real usage.

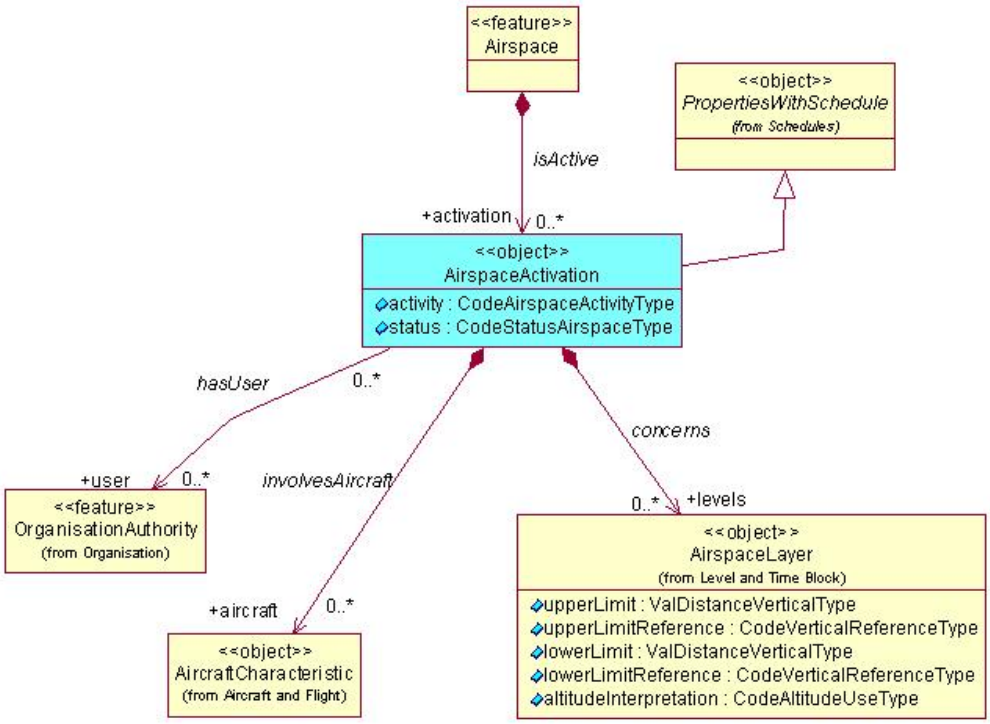
Modify the class LayerAndTime as follows:

- new name = AirspaceLayer
- definition = " *The portion of airspace between two specified vertical levels*."
- remove the inheritance association from PropertiesWithSchedule (introduced by Change 5.1-35)

Declare the following classes as specialisations of PropertiesWithSchedule class (in order to compensate the removal of the schedule from AirspaceLayer):

- AirspaceLayerClass

The resulting AirspaceActivation model is displayed in the diagram below:



RouteAvailability

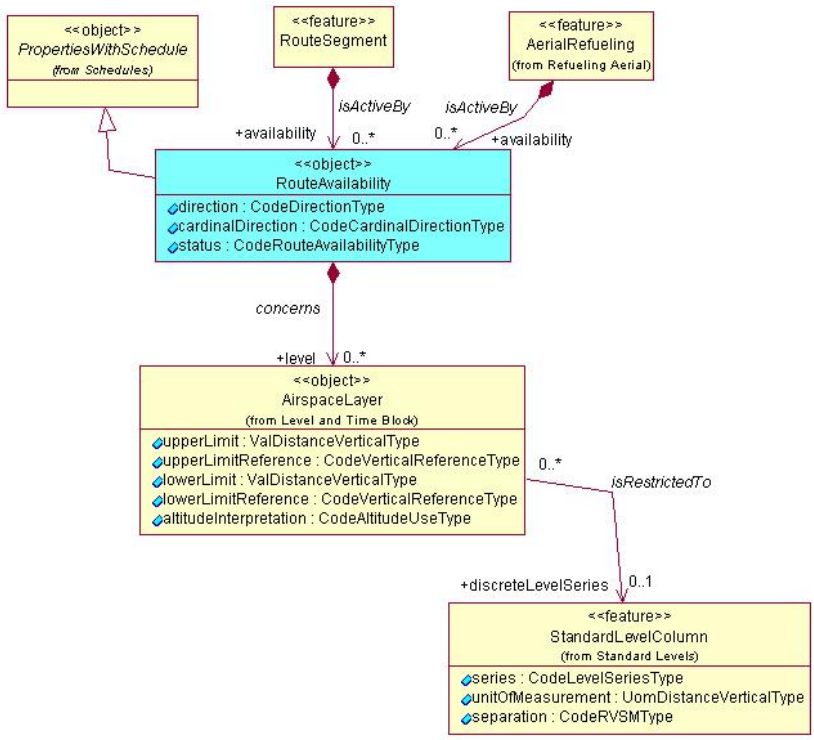
In the Routes package of the UML model, rename the diagram "5 - Route Availability" instead of "5 - Route Portion Usage"

Modify the class RoutePortionUsage as follows:

- rename as RouteAvailability
- change stereotype into "object"
- definition = " *Information about the operational availability of a route*"
- declare the RouteAvailability class as a specialisation of the PropertiesWithSchedule
- move the following attributes from RouteAvailability (former RoutePortionUsage) into the RouteSegment class:
  - minimumEnrouteAltitude
  - minimumCrossingAtEnd
  - minimumCrossingAtEndReference
  - maximumCrossingAtEnd
  - maximumCrossingAtEndReference
  - navigationType
  - requiredNavigationPerformance
  - designatorSuffix
- insert a new attribute "status", " *The availability status of the route, in the direction indicated by the direction and/or cardinalDirection attribute*", data type CodeRouteAvailabilityType
- delete the association with RoutePortion
- new associations:
  - RouteAvailability concerns [0..*] LayerAndTime (role=" *levels*", aggregated by value), navigable towards the LayerAndTime class
  - RouteSegment isActiveBy [0..*] RouteAvailability (role=" *availability*", aggregated by value), navigable towards the RouteAvailability class
  - AerialRefueling isActiveBy [0..*] RouteAvailability (role=" *availability*", aggregated by value), navigable towards the RouteAvailability class

Delete the class RoutePortionUsageLimitation. Delete the class AerialRefuelingUsage

The resulting RouteAvailability model is displayed in the diagram below:

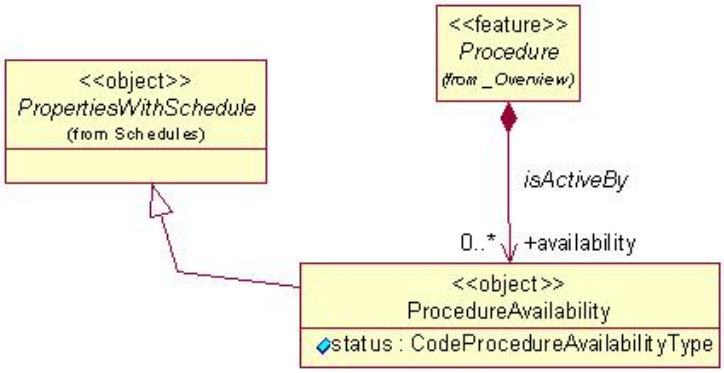


ProcedureAvailability

Modify the class ProcedureUsage as follows:

- rename into ProcedureAvailability
- change stereotype into "object"
- delete association with Timetable
- declare as a specialisation of the PropertiesWithSchedule class
- delete the association with Procedure
- insert a new association: Procedure isActiveBy [0..*] ProcedureAvailability (role=" *availability*", aggregated by value), navigable towards the ProcedureAvailability class
- rename the attribute "usageStatus" into "status".

The resulting ProcedureAvailability model is displayed in the diagram below:



AirportHeliport - UsageCondition

Modify the AirportHeliportUsageLimitation as follows:

- rename it as "UsageCondition";
- declare it "abstract";
- new definition: " *A rule governing the usage of an Airport/Heliport or of one of its surfaces*";
- remove the "reason" attribute;
- remove all its existing associations;
- insert a new association: UsageCondition hasContactInstructions [0..*] ContactInformation (role=" *contact*", aggregated by value), navigable towards the ContactInformation class.

Insert a new class "object" Meteorology as follows:

- definition = " *A specific set of meteorological conditions*";
- attributes (most copied from the previous AirportHeliportUsageCondition class):
  - flightConditions = " *An overall indication whether the meteorological conditions allow visual or instrumental operations*", data type CodeMeteoConditionsType;
  - visibility = " *The greatest distance at which lights of 1,000 candelas can be seen and identified against an unlit background (as reported by the airport)*", data type ValDistanceType;
  - visibilityInterpretation = " *Indicates whether the visibility is a minimum or a maximum value*", data type CodeValueInterpretationType;
  - runwayVisualRange = " *The distance over which a pilot of an aircraft on the centreline of the runway can see the runway surface markings or the lights delineating the runway or identifying its centre line*", data type ValDistanceType;
  - runwayVisualRangeInterpretation = " *Indicates whether the RVR is a minimum or maximum value*", data type CodeValueInterpretationType;

Insert a new class "object" ConditionCombination as follows:

- definition = " *A set of filter criteria used to determine the subset of flights, environmental conditions and times for which the usage is specified*";
- declare it as a specialisation of the PropertiesWithSchedule class;
- attributes:
  - logicalOperator = " *The type of operation combining an elementary condition or a previously defined combination with other elementary conditions or previously defined combinations*", data type CodeLogicalOperatorType
- associations:
  - UsageCondition appliesToSelection [0..1] ConditionCombination (role=" *selection*", aggregated by value), navigable towards the ConditionCombination class.
  - ConditionCombination dependsOnWeather [0..*] Meteorology (role=" *weather*", aggregated by value), navigable towards the Meteorology class;
  - ConditionCombination dependsOnAircraft [0..*] AircraftCharacteristic (role=" *aircraft*", aggregated by value), navigable towards the AircraftCharacteristic class;
  - ConditionCombination dependsOnFlight [0..*] FlightCharacteristic (role=" *flight*", aggregated by value), navigable towards the FlightCharacteristic class;
  - ConditionCombination dependsOnCombination [0..*] ConditionCombination (role=" *subCondition*", aggregated by value), navigable towards the ConditionCombination class.

Delete the AirportHeliportUsageCondition class

~~Insert a new "choice" ConditionElementChoice class as follows: * definition = " *A link class, allowing to select between an elementary condition and a preceding combination of conditions*";~~

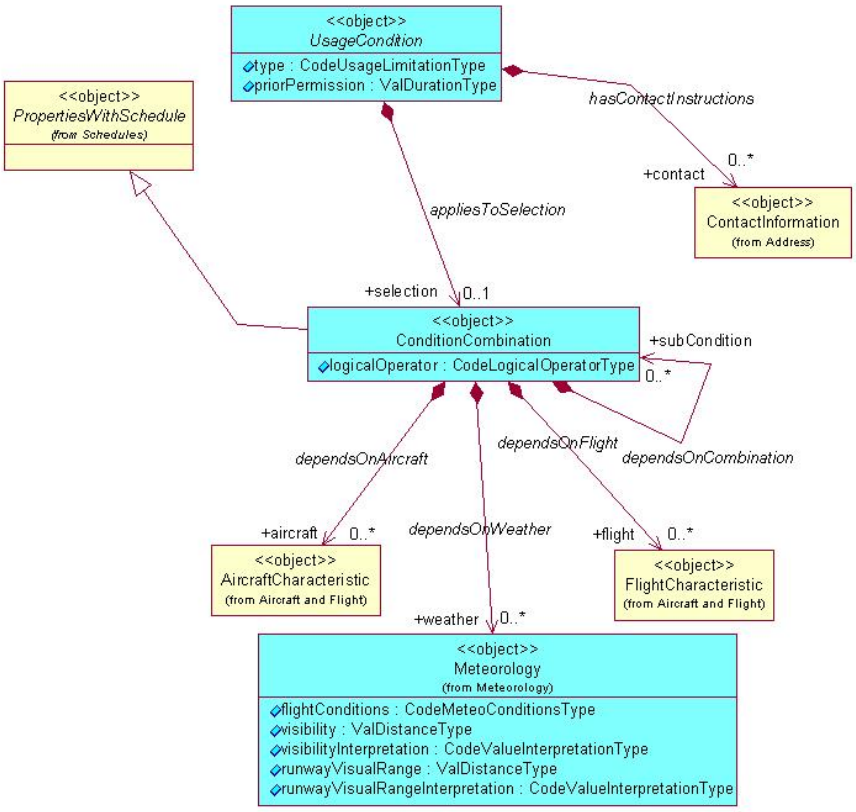
- ~~associations~~
  - ~~ConditionCombination hasOperand [0..*] ConditionElementChoice (role=" *operand*", aggregated by value), navigable towards the ConditionElementChoice class;~~
  - ~~ConditionElementChoice isWeather [0..1] Meteorology (role=" *weather*", aggregated by value), navigable towards the Meteorology class;~~
  - ~~ConditionElementChoice isAircraft [0..1] AircraftCharacteristic (role=" *aircraft*", aggregated by value), navigable towards the AircraftCharacteristic class;~~
  - ~~ConditionElementChoice isFlight [0..1] FlightCharacteristic (role=" *flight*", aggregated by value), navigable towards the FlightCharacteristic class;~~
  - ~~ConditionElementChoice isCombination [0..*] ConditionCombination (role=" *subCondition*", aggregated by value), navigable towards the ConditionCombination class.~~

Insert a new "enumeration" data type class "CodeLogicalOperatorType"

- definition = " *A boolean operator*"
- enumerated list of values:
  - AND = " *The result is true only when all operands are true*"
  - OR = " *The result is true if any operand is true*"

- NOT = " *The result is the opposite of the operands true/false state (it applies to a single operand)*"
- NONE = " *No operation (used for single conditions)*"

The resulting AirportHeliportUsage model is displayed in the diagram below):



AirportHeliportAvailability

In the sub-package Airport/Heliport, create a new diagram "AirportHeliport Availability"

Create a new "object" class AirportHeliportAvailability as follows:

- definition = " *Information about the operational status of the airport/heliport*"
- declare it as specialisation of the PropertiesWithSchedule
- new association AirportHeliport isOperationalBy [0..*] AirportHeliportAvailability (role=" *availability*", aggregated by value) , navigable towards the AirportHeliportAvailability class
- attributes:
  - operationalStatus = " *Indicates the availability of the facility for specific flight operations.*", data type CodeStatusAirportType
  - warning = "~~A reason for caution when operating at the facility", data type CodeAirportWarningType

Create a new "enumeration" CodeStatusAirportType

- definition = " *A coded list of values that indicates the availability of an airport/heliport facility for specific flight operations*"
- enumerated list of values:
  - NORMAL = The facility operates with nominal parameters
  - LIMITED = The facility operates below its nominal parameters, with additional usage restrictions
  - CLOSED = The facility is not operational
  - OTHER = Other.

Create a new "object" class AirportHeliportUsage as follows:

- definition = " *A rule governing the usage of an AirportHeliport*"
- declare it as specialisation of the UsageCondition

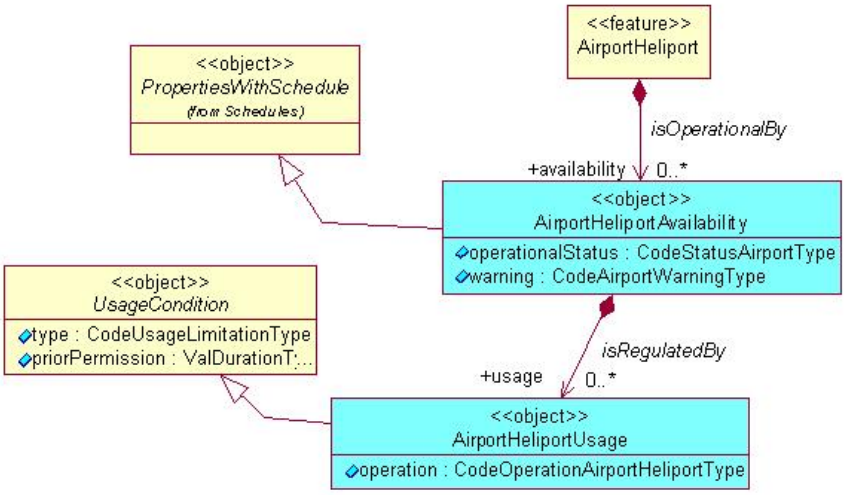
- attributes:
  - operation = " *A type of activity for which a usage rule is specified*", data type CodeOperationAirportHeliportType
- associations:
  - AirportHeliportAvailability isRegulatedBy [0..*] AirportHeliportUsage (role=" *usage*", aggregated by value) , navigable towards the AirportHeliportUsage class

Modify the CodeAirportHeliportOperationType as follows:

- rename as CodeOperationAirportHeliportType
- remove the value "TAXIING"

Delete the AirportHeliportOperation class

The resulting AirportHeliportAvailability model is displayed in the diagram below):



ManoeuvringAreaAvailability

In the sub-package AirportHeliport, create a new diagram "Manoeuvring Area Availability"

Create a new "object" class ManoeuvringAreaAvailability as follows:

- definition = " *Information about the operational status of the manoeuvring area elements (runway, taxiway, TLOF, etc.)*"
- declare it as specialisation of the PropertiesWithSchedule
- associationa:
  - RunwayDirection isOperationalBy [0..*] ManoeuvringAreaAvailability (role=" *availability*", aggregated by value), navigable towards the ManoeuvringAreaAvailability class
  - RunwayElement isOperationalBy [0..*] ManoeuvringAreaAvailability (role=" *availability*", aggregated by value), navigable towards the ManoeuvringAreaAvailability class
  - Taxiway isOperationalBy [0..*] ManoeuvringAreaAvailability (role=" *availability*", aggregated by value), navigable towards the ManoeuvringAreaAvailability class
  - TaxiwayElement isOperationalBy [0..*] ManoeuvringAreaAvailability (role=" *availability*", aggregated by value), navigable towards the ManoeuvringAreaAvailability class
  - TouchDownLiftOff isOperationalBy [0..*] ManoeuvringAreaAvailability (role=" *availability*", aggregated by value), navigable towards the ManoeuvringAreaAvailability class
  - SeaplaneLandingArea isOperationalBy [0..*] ManoeuvringAreaAvailability (role=" *availability*", aggregated by value), navigable towards the ManoeuvringAreaAvailability class
- attributes:
  - operationalStatus = " *Indicates the availability of the facility for specific flight operations.*", data type CodeStatusAirportType
  - warning = "~~A reason for caution when operating at the facility", data type CodeAirportWarningType

Create a new "object" class ManoeuvringAreaUsage as follows:

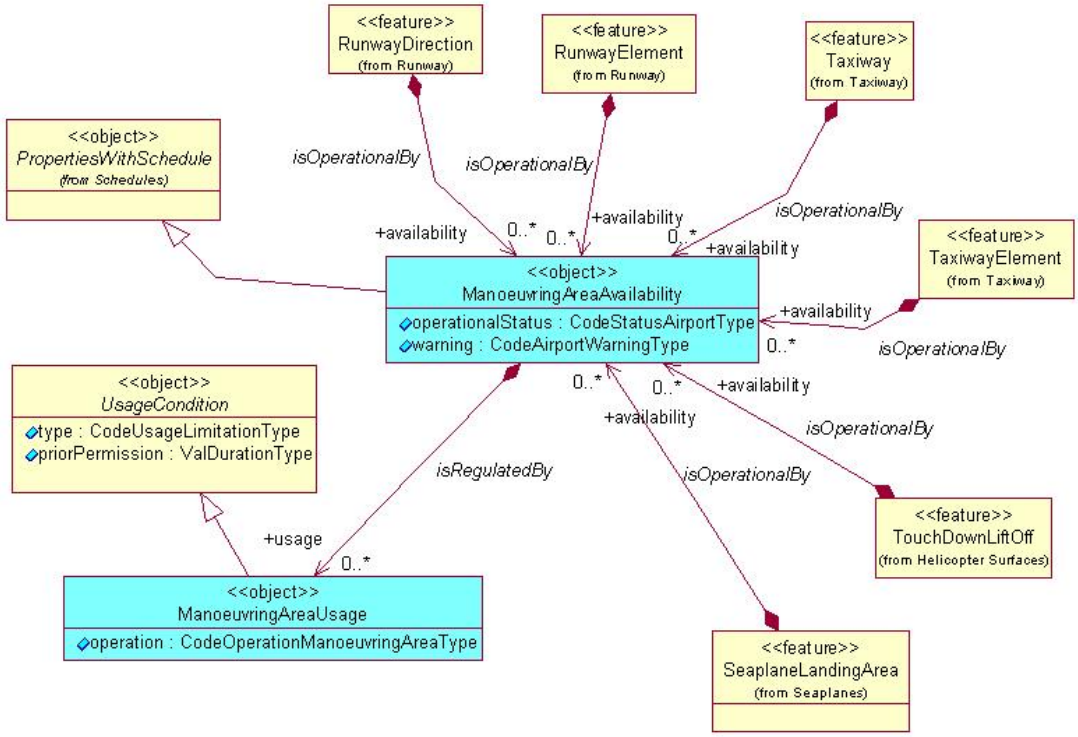
- definition = " *A rule governing the usage of the manoeuvring area element*"

- declare it as specialisation of the UsageCondition
- attributes:
  - operation = " *A type of activity for which a usage rule is specified*", data type CodeOperationManoeuvringAreaType
- associations:
  - ManoeuvringAreaAvailability isRegulatedBy [0..*] ManoeuvringAreaUsage (role=" *usage*", aggregated by value) , navigable towards the ManoeuvringAreaUsage class

Create a new "enumeration" CodeOperationManoeuvringAreaType data type as follows:

- definition = "A coded list of values that indicates a type of activity on an airport/heliport manoeuvring area element"
- list of values:
  - LANDING = " *Aircraft landing*"
  - TAKEOFF = " *Aircraft take off*"
  - TOUCHGO = " *Aircraft landing followed by an immediate take-off, without deceleration*"
  - TRAIN_APPROACH = " *Aircraft practices low approaches*"
  - TAXIING = " *Aircraft taxiing along an element of the airport/heliport movement area*"
  - CROSSING = " *Crossing an element of the airport/heliport manoeuvring area, as opposed to taxiing along that element*"
  - AIRSHOW = " *Airshow and aircraft display activities*"
  - ALL = " *All operation types*"
  - OTHER = "Other"

The resulting ManoeuvringAreaAvailability model is displayed in the diagram below):



ApronAreaAvailability

In the sub-package AirportHeliport, create a new diagram "Apron Area Availability"

Create a new "object" class ApronAreaAvailability as follows:

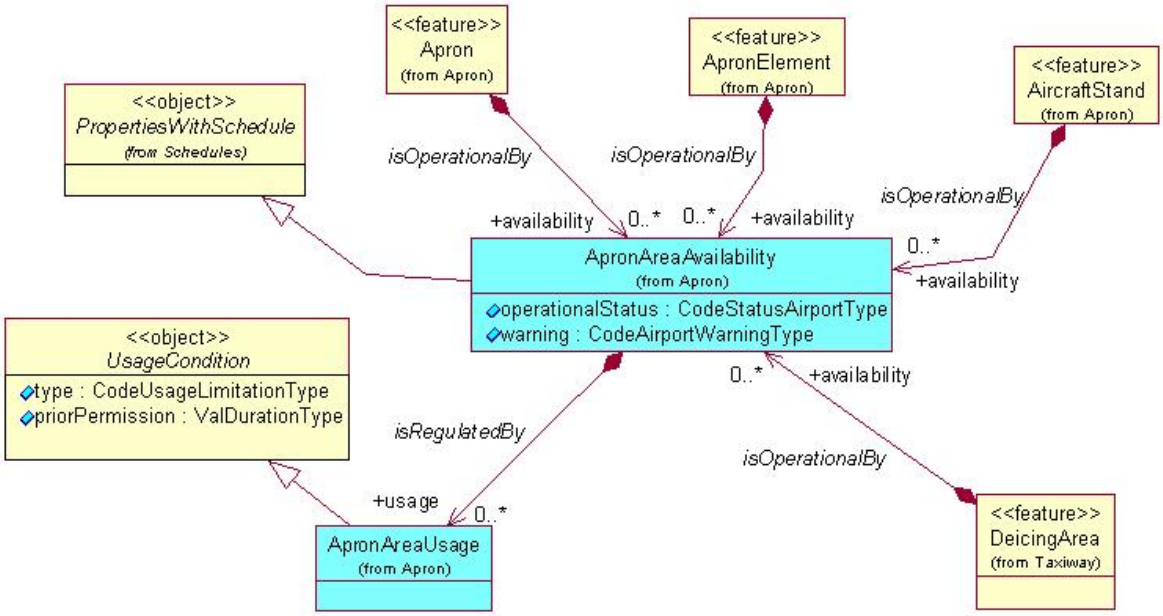
- definition = " *Information about the operational status of an element situated in the apron area*"
- declare it as specialisation of the PropertiesWithSchedule
- associations:

    - Apron isOperationalBy [0..*] ApronAreaAvailability (role=" *availability*", aggregated by value), navigable towards the ApronAvailability class
    - ApronElement isOperationalBy [0..*] ApronAreaAvailability (role=" *availability*", aggregated by value), navigable towards the ApronAvailability class
    - AicraftStand isOperationalBy [0..*] ApronAreaAvailability (role=" *availability*", aggregated by value), navigable towards the ApronAvailability class
    - DeicingArea isOperationalBy [0..*] ApronAreaAvailability (role=" *availability*", aggregated by value), navigable towards the ApronAvailability class
- attributes:
    - operationalStatus = " *Indicates the availability of the facility for specific flight operations.*", data type CodeStatusAirportType
    - warning = "~~A reason for caution when operating at the facility", data type CodeAirportWarningType

Create a new "object" class ApronAreaUsage as follows:

- definition = " *A rule governing the usage of an element situated in the apron area*"
- declare it as specialisation of the UsageCondition
- associations:
    - ApronAreaAvailability isRegulatedBy [0..*] ApronAreaUsage (role=" *usage*", aggregated by value) , navigable towards the ApronAreaUsage class

The resulting ApronAreaAvailability model is displayed in the diagram below):



Data type of 'status' attributes

Delete the status attribute of the following classes:

- Apron
- DeicingArea
- TouchDownLiftOff

Change the data type of the "status" attribute into CodeStatusOperationsType (instead of CodeStatusSurfaceType) in the following classes:

- ArrestingGear
- Road
- RunwayBlastPad
- RunwayProtectArea
- TaxiHoldingPosition

Delete the CodeStatusSurfaceType data type.

'Abandoned' attribute

Insert an 'abandoned' attribute (" *Indicating that the surface is no longer in operational use, but it is still physically present and visible, although usually in a degraded state*", data type "CodeYesNoType") into the following classes:

- Runway
- Apron
- Taxiway
- TLOF
- Road

FlightRestriction

Insert new associations:

- FlightConditionElementChoice isWeather [0..1] Meteorology (role=" *weather*", aggregated by value), navigable towards the Meteorology class;
- FlightConditionElementChoice isAerialRefueling [0..1] AerialRefueling (role=" *aerialRefuelingCondition*", aggregated by value), navigable towards the AerialRefueling class;
- FlightRoutingElementChoice isAerialRefueling [0..1] AerialRefueling (role=" *aerialRefuelingElement*", aggregated by value), navigable towards the AerialRefueling class;
- FlightRestrictionRoute hasContactInstructions [0..*] ContactInformation (role=" *contact*", aggregated by value), navigable towards the ContactInformation class.

Insert a new attribute into the FlightRestrictionRoute

- priorPermission = " *The minimum lead time required for contacting the relevant authority in order to get permission for using the routing*"