**ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE**

**Fakulta Elektrotechnická**

**Katedra Kybernetiky**

# Reprezentace dat AIXM 5.1 v relační databázi

# AIXM 5.1 data representation in a relational database

Diplomová práce

Studijní program: Otevřená informatika
Studijní obor: Softwarové inženýrství

Vedoucí práce: doc. Ing. Zdeněk Kouba CSc.

**Bc. Jan Šrogl**

**Praha rok 2016**

# <Zadání práce>

Tímto bych chtěl poděkovat svému vedoucímu diplomové práce panu doc. Ing. Zdeňkovi Koubovi CSc. za odborné vedení při vypracování této práce. Dále bych chtěl mnohokrát poděkovat své rodině za obětavou morální podporu při tvorbě a finalizování této práce.

Prohlašuji, že jsem svou diplomovou práci na téma Reprezentace dat AIXM 5.1 v relační databázi napsal samostatně a výhradně s použitím citovaných pramenů.

V Praze dne: 27.5.2016

……...

Bc. Jan Šrogl

## Abstrakt

Tato práce se zabývá možnostmi reprezentace dat AIXM 5.1 v relační databázi. Zpočátku je pojednáno o principu reprezentace geometrické informace v obecné rovině. Jsou zde rovněž uvedeny příklady souřadnicových systémů. V závěru počátečních kapitol jsou uvedeny příklady geometrických jazyků včetně jazyků GML a AIXM. Následné kapitoly hlouběji rozebírají podstatu jazyků GML a AIXM. V případě jazyka GML je podrobně popsána struktura geometrií. V případě jazyka AIXM je představen celkový model tohoto jazyka. Následující část práce se pak zabývá mapováním mezi geometriemi z Microsoft SQL Serveru Spatial a jazyka GML. Kapitola nejprve představuje jednotlivé geometrické datové typy SQL Serveru a následně je srovnává s geometriemi GML. Závěrem této části je uvedeno možné mapování mezi SQL Serverem a geometriemi jazyka GML. Druhá, spíše praktičtější, část práce pojednává o možnostech tvorby modulu pro import dat. Nejprve jsou vytipovány použitelné technologie. Dále je předložen návrh možné struktury programu. Nakonec je uveden popis implementace. Závěrem práce je pak uveden záznam o měření funkčnosti a možné vize do budoucna.



## Klíčová slova

AIXM 5.1, SQL Server, GML, Spatial database



## Abstract

This thesis aims on possibilities of AIXM 5.1 data representation in the relational database. Firstly there's a discussion about basic principles of geometry information and its representation. There are also some examples of coordinate systems. In conclusion of those chapters, there are some examples of geographical languages including GML and AIXM languages. Next chapters describe the principles of GML and AIXM languages in depth. In case of GML, the structure of geometries is being described. In case of AIXM language, the model of this language is being introduced. Next part of the thesis deals with the mapping between geometries from Microsoft SQL Server Spatial and the GML language. The part firstly introduces the individual data types of the SQL Server, which are then being compared to GML geometries. In the conclusion of this part, there's introduced some mapping between SQL Server datatypes and GML geometries. The second, more practical part discusses the possibilities of making a module for data import. Firstly, some technologies for use are being identified. Furthermore, the possible structure of the program is being proposed. Finally there are some descriptions of module's implementation. At the end of the thesis, the notes of functionality checking process are being quoted, including some visions for the future development.



## Keywords

AIXM 5.1, SQL Server, GML, Spatial database

# Seznam zkratek

| | |
|---|---|
| GML | Geography Markup Language |
| AIXM | Aeronautical Information Exchange Model |
| ORM | Objektově relační mapování |
| WKT | Well-Known Text |
| SRID | Spatial reference ID |
| OGC | Open Geospatial Consorcium |

# Obsah

Abstrakt ..... 4
Klíčová slova ..... 4
Abstract ..... 4
Keywords ..... 4
Seznam zkratek ..... 5
Úvod ..... 9
1. Reprezentace geometrické informace ..... 9
1.1 Souřadné systémy ..... 9
1.1.1 WGS84 ..... 9
1.1.2 CRS84 ..... 9
1.1.3 ETRS89 ..... 10
1.1.4 SRID identifikátor ..... 10
1.2 Typické základní útvary ..... 10
1.3 Reprezentace prostorových vztahů ..... 11
1.3.1 9-Intersection Model ..... 11
1.4 Příklady geometrických jazyků ..... 12
1.4.1 GeoJSON ..... 12
1.4.2 Well-known Text ..... 12
1.4.3 Simple Features ..... 13
1.4.4 GML ..... 13
1.4.5 AIXM ..... 13
2. Jazyk GML ..... 13
2.1 Účel jazyka ..... 13
2.2 Základní členění geometrií (knihovny) ..... 14
2.3 Základní geometrické útvary ..... 14
2.4 Rozšiřující útvary ..... 14
2.5 Koncept Feature a Property ..... 16
2.6 Projekční systémy ..... 16
2.7 Zápis geometrických souřadnic ..... 16
3. AIXM 5.1. ..... 17
3.1 Základní koncepce jazyka ..... 17
3.2 Prvek Feature ..... 18
3.3 Temporální model ..... 18
3.3.1 Trvalé a dočasné změny ..... 20
3.3.2 Verzování atributů Feature ..... 21

3.3.3 Korekce, změny hodnot u Feature ..... 21
3.4 Reference v AIXM 5.1 ..... 22
3.4.1 Konkrétní reference v rámci datové sady ..... 22
3.4.2 Konkrétní externí reference řešené pomocí webových služeb ..... 22
3.4.3 Abstraktní reference, které mají být vyřešeny externí aplikací ..... 22
3.4.4 Postup při řešení reference ..... 23
3.5 Prvek Object ..... 23
3.6 Systém geometrií v AIXM 5.1 ..... 23
3.7 AIXM 5.1 dokument ..... 24
4. Srovnání geometrií se systémem MS SQL Server Spatial ..... 25
4.1 Geometrie v Microsoft SQL server spatial ..... 25
4.2 Podobnosti datových struktur s GML 3.2 ..... 26
4.3 Mapování geometrií mezi GML a SQL Server Spatial ..... 27
4.3.1 Mapování geometrií z SQL Server Spatial do GML ..... 27
4.3.2 Mapování geometrií z GML do SQL Server Spatial ..... 28
5. Návrh datového modelu pro SQL server ..... 28
5.1 Klíčové vlastnosti standardu AIXM 5.1 ..... 28
5.1.1 Řešení referencí v AIXM 5.1 ..... 29
5.2 Ukládání XML dat do databáze ..... 29
5.2.1 Databáze nad systémem souborů ..... 29
5.2.2 Relační databáze ..... 29
5.3 Návrh datového modelu ..... 30
5.3.1 Základní entity ..... 30
5.3.2 Rozšíření entity Property ..... 32
5.3.3 Entita GeometryProperty ..... 32
5.4 Souřadnicové systémy v SQL Server Spatial ..... 32
6. Modul pro import a export AIXM 5.1 dat ..... 33
6.1 Přehled existujících technologií ..... 34
6.1.1 Parsování XML dat ..... 34
6.1.2 ORM ..... 35
6.1.3 Parsování Geometrií do Microsoft SQL Server Spatial ..... 36
6.2 Návrh modulu ..... 37
6.2.1 Parsování dat ..... 37
6.2.2 ORM ..... 38
6.2.3 Dotazování ..... 38
6.3 Implementace ..... 40

6.3.1 Parsování Dat ..... 40
6.3.2 Datový model ..... 41
6.3.3 Reprezentace geometrické informace ..... 41
6.3.4 Detekce projekce ..... 41
7. Ověření funkčnosti modulu ..... 42
7.1 Testovací data ..... 42
7.2 Způsob měření ..... 42
7.3 Výsledky měření ..... 42
8. Rozvoj do budoucna ..... 43
9. Závěr ..... 43
Seznam literatury ..... 44
Seznam obrázků ..... 47
Seznam tabulek ..... 48
Seznam kódů ..... 49
Obsah CD ..... 50

# Úvod

V letecké dopravě se setkáváme s množstvím informací, které je třeba ukládat. Důležitou kategorií jsou geografické informace o pozicích jednotlivých objektů na povrchu, či v povětří. Standard AIXM 5.1 přináší možnost robustního přenosu těchto dat napříč aplikacemi. Tato data je nutné nějakou rozumnou formou uchovávat a pružně měnit. Následující kapitoly pojednávají o podstatě geometrické informace a možností její reprezentace. Následně budou uvedeny příklady geometrií v jazyce GML, na kterém je jazyk AIXM 5.1 založen. Dále budou geometrie jazyka GML podrobeny srovnání s datovými typy SQL Serveru Spatial. Toto srovnání je zakončeno návrhem možného mapování mezi jednotlivými geometriemi. Další část práce se zaměřuje na analýzu technologií a návrh modulu pro import dat. Závěrem práce je popsána tvorba tohoto modulu a záznam o ověření funkčnosti tohoto modulu. Na konec jsou nadneseny možné směry dalšího rozvoje modulu.

# 1. Reprezentace geometrické informace

Tato kapitola pojednává o možných přístupech a standardech k reprezentaci geometrické informace. Geometrická informace je zpravidla množina různých geometrických útvarů, které reprezentují nějaký objekt. V reálném prostředí to může být například most, pole, letištní dráha, heliport apod., nemusíme se však omezovat pouze na hmatatelné objekty. Dosti běžným jevem je například uchovávání informace o povětrnostních podmínkách v lokalitě (průběhy front) či mapování oblasti výskytu sopečného popelu v ovzduší. Pro účely letecké dopravy se rovněž používá mapování křivek přistávacích a vzletových koridorů, či vzdušných tras.

Takovýto útvar, kromě popisu svého účelu, obsahuje také informaci o jeho poloze v prostoru. Poloha objektu v prostoru je vyjádřena v souřadném systému prostoru. V praxi se můžeme setkat s použitím *planárních souřadnic* (projekce na plochu), případně *sférických souřadnic* (projekce na kouli). V případě sférického zobrazení, vzhledem k neideálnímu tvaru Země, vzniklo množství projekčních systémů. Tyto projekční systémy se povětšinou liší v definici zakřivení zemského povrchu, je však možné setkat se také s projekčními systémy, které vyjadřují geografickou informaci ve specifickém formátu či pořadí geografických souřadnic.

## 1.1 Souřadné systémy

Vzhledem k průběžnému pokroku v přesnosti měření zemského povrchu vzniklo v průběhu času mnoho různých souřadných systémů. Tato kapitola pojednává o vybraných souřadných systémech.

### 1.1.1 WGS84

„Světový geodetický systém (World Geodesic System) je standard používaný v kartografii, geodézii a navigačních systémech, včetně GPS. Využívá standardní souřadnicový systém Země, standardní sférickou referenci povrchu pro data v surové nadmořské výšce a gravitační ekvipotencionální plochu (geoid), která definuje nominální hladinu moře.“ [9] Tento souřadný systém je také znám pod označením WGS1984 či EPSG:4326. Výhodou tohoto systému je velká podpora v mnoha aplikacích, geografických systémech a také databázových aplikacích. Geografická informace je reprezentována ve formátu zeměpisná šířka – zeměpisná délka (angl. Latitude – Longitude).

### 1.1.2 CRS84

Jedná se o mutaci souřadného systému, který vychází ze souřadného systému WGS84. Rozdílem oproti systému WGS84 však je rozdílná reprezentace geografických souřadnic. Zatímco systém WGS84 reprezentuje souřadnice v pořadí zeměpisná šířka – zeměpisná délka (angl. Latitude – Longitude), souřadnicový systém CRS84 reprezentuje tyto souřadnice v opačném pořadí, tedy

zeměpisná délka – zeměpisná šířka (Longitude – Latitude). Tento standard je vlastním standardem OGC. [11]

### 1.1.3 ETRS89

„Evropský pozemní referenční systém 1989 (European Terrestrial Reference System 1989) je tzv. ECEF (Earth–Centred, Earth–Fixed) geodetický, kartézský referenční systém, v němž je Euro–Asijská deska jako celek statická. Souřadnice a mapy zanesené v referenčním systému ETRS89 jsou imunní vůči posunu tektonických desek.” [12]

### 1.1.4 SRID identifikátor

Pro potřeby prostorových databází (Spatial database), geografických informačních systémů (GIS) a obecně aplikací pracujících s prostorovými daty byl vytvořen identifikátor, který jednoznačně identifikuje, v jakém projekčním systému jsou data reprezentována. „Spatial reference ID (SRID) je identifikátor, který specifikuje, ve kterém elipsoidním souřadném systému je instance Geography reprezentována.“

Kupříkladu projekční systém ETRS89 má SRID 4258, hojně užívaný systém WGS84 pak 4326. Tyto souřadné systémy spravuje několik asociací, z nichž možno jmenovat EPSG (European Petroleum Survey Group ) a OGC (Open Geospatial Consorcium). [7] [12]

## 1.2 Typické základní útvary

V drtivé většině reprezentací je zpravidla možné setkat se s geometrickým typem *bod* a *přímka (*v GML, GeoJSON a SQL Server Spatial nazýván jako *Point* a *LineString*), případně jejich množinami. Takovýto bod lze použít například k jednoduchému označení místa, kde se na mapě nachází významný objekt, jako například stanoviště pozemního radaru, či pozice letištní věže. V případě přímky pak lze namapovat, kupříkladu, základní orientaci vzletové a přistávací dráhy. Při použití většího počtu bodů pro popis složitějšího geografického útvaru je možné ukládat tyto body samostatně, v takovémto případě se však ztratí informace o celkovém společném významu těchto bodů. V takovýchto případech může být lepší použití datového typu představujícího kolekci takovýchto geometrií. Kolekce bodů se vyskytuje v mnoha geografických jazycích, převážně pod jmény MultiPoint. Pro ukládání množiny přímek je také lepší zvolit obdobnou reprezentaci. Kolekce přímek je však možné, z hlediska struktury, rozdělit na dvě množiny, nezávislé vzhledem k vnitřnímu uskupení přímek. Jednou takovou množinou je kolekce přímek, v níž každá přímka má maximálně dva společné koncové body s ostatními přímkami (jednotlivé přímky jsou na sebe napojeny a tvoří dohromady jakousi lomenou čáru). Druhá množina přímek pak umožňuje vložení libovolně orientovaných přímek a nevyžaduje, aby byly tyto přímky propojeny. V GML je možné pro první zmíněnou kolekci bodů použít datového typu MultiPoint, pro kolekci vzájemně propojených přímek pak jazyk GML disponuje datovým typem MultiLineString, v SQL Server Spatial je pro reprezentaci množiny bodů použit datový typ MultiPoint, pro kolekci vzájemně provázaných přímek pak SQL Server Spatial disponuje datovým typem LineString, který sám o sobě reprezentuje kolekci jedné a více přímek, pro kolekci libovolně orientovaných přímek disponuje datovým typem MultiLineString.

Z takovýchto útvarů lze již obstojně namodelovat středně složité útvary. V některých případech je však nutné pracovat s daleko přesnějšími daty. Je zřejmé, že pro modelování kruhových ploch či modelování matematicky definovaných křivkových útvarů dojde, při použití doposud jmenovaných geometrických útvarů, ke ztrátě přesnosti. V takovémto případě je nutné tuto ztrátu minimalizovat nějakou rozumnou aproximací. Nevýhodou však stále zůstává ne zcela ideální průběh útvaru oproti zamýšlenému originálu. Rovněž při použití takovýchto aproximací roste počet prvků, které musíme ukládat, což se může negativně projevit na velikosti výsledné datové struktury (databáze).

Z tohoto důvodu většina geometrických jazyků disponuje složitějšími datovými typy, jako jsou například datové typy reprezentující složitější křivkové útvary, či datové typy reprezentující ohraničené plochy. Složitější křivkové útvary je možné reprezentovat pomocí geometrickýh typů, souhrnně nazývaných jako *křivka*. Jedná se o již složitější geometrický útvar, který se může skládat z jednoho či více základních útvarů. Tyto útvary jsou na sebe (stejně jako v případě lomené čáry) vzájemně navázány a dohromady tvoří zamýšlený útvar. Nejjednoduššími objekty v křivce může být již zmiňovaná kolekce přímek, pro zjemnění průběhu však může křivka obsahovat také nelineární geometrické objekty – pokud to prostředí jazyka dovoluje, je možné křivku složit například z kruhových křivek. V jazyce GML je možné setkat se se souhrnným abstraktním typem Curve, který zastřešuje veškeré složitější tvary. SQL Server Spatial pak disponuje množstvím reprezentací takovýchto útvarů, z velké škály je možné jmenovat datový typ CircularString pro kruhovité křivky, či typ CompoundCurve, který kombinuje kruhovité křivky s lomenými čarami.

Takovéto geometrické typy již mohou obstojně reprezentovat libovolný geometrický útvar. Možnou nevýhodou však stále zůstává množství dat, které je třeba k reprezentaci geometrické informace uchovávat. Pro případ modelování geometrických útvarů, které jsou podobné geometrickým funkcím, je lepší reprezentovat takovéto tvary pomocí pokročilejších útvarů, jako jsou *Splajny*, či *Bezierovy křivky*. Jazyk GML pro takovýto případ disponuje podmnožinou typu Curve, obsahující datové typy CubicSpline, BezierCurve a Bspline.

Dalším rozšířením křivkových útvarů může být podpora geometrických typů reprezentující plochu. Takovýto útvar by šlo lehce namodelovat za pomocí již zmíněných geometrií. V praxi by však mohlo docházet k nejasnostem, zdali příslušný útvar má být chápán jako uzavřená křivka, či uzavřená plocha. Jazyk GML pro tyto případy disponuje datovým typem Polygon. SQL Server Spatial pak nabízí datové typy Polygon a CurvePolygon.

Doposud jsme v této kapitole rozebírali pouze základní útvary, které lze zobrazit do dvourozměrného prostoru. Nutno poznamenat, že některé ze zmíněných datových útvarů samozřejmě umožňují přidání souřadnice třetího rozměru, z hlediska přehlednosti je však třeba přistupovat k trojrozměrným geometriím z jakéhosi nadhledu v podobě objektů. V případě potřeby modelování rozsáhlých trojrozměrných nelineárních ploch je tedy třeba dalších datových typů. V GML je možné tohoto dosáhnout za pomoci abstraktních datových typů Surface a Patch. SQL Server Spatial sám o sobě speciálními trojrozměrnými typy nedisponuje, je však možné vybrat si z některé ze zastřešujících geometrií, jako například GeometryCollection, či FullGlobe.

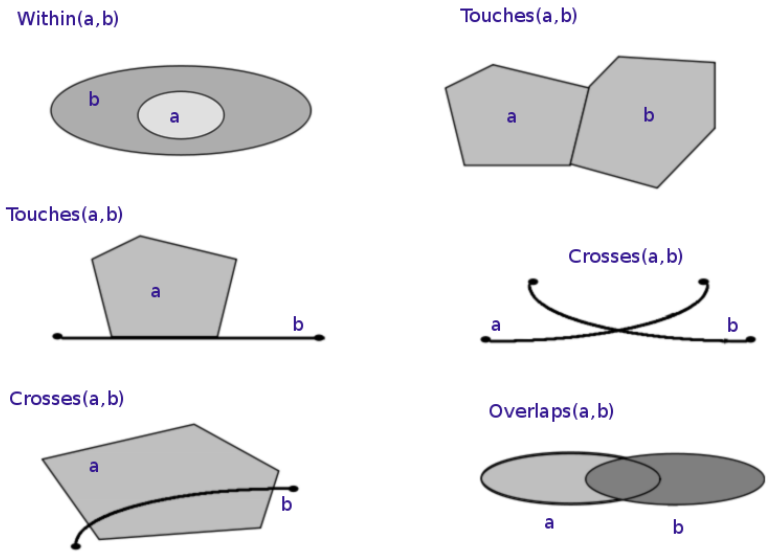
## 1.3 Reprezentace prostorových vztahů

Další kapitolou je reprezentace vztahů mezi různými geometrickými útvary. Zde přichází na řadu problém protínání a překrývání útvarů. Jedním ze standardů řešících tuto problematiku v abstraktní rovině je 9-Intersection model.

### 1.3.1 9-Intersection Model

„Dimensionally Extended nine-Intersection Model (zkráceně DE-9IM) je topologický model a standard používaný pro popis prostorových vztahů dvou regionů (dvou geometrických útvarů ve 2D prostoru R^2).“ [16] Tento model se zabývá definicemi všech možných pozic dvou útvarů. Zavádí pojem vzájemné pozice dvou geometrických útvarů, která je jednoznačně definována a neměnná vůči případnému zvětšení, projekcím či rotaci. Na základě pokusů [15], [17] bylo zjištěno 512 různých možností pozic dvou geometrických útvarů. Tyto pozice je možné zakódovat do matice 3x3, která reprezentuje unikátní typ vzájemné polohy dvou těles. Takovýmto způsobem je tedy možné popsat

vzájemné polohy všech geometrických útvarů. Z tohoto standardu, kromě množství dalších pozic, vyplývají základní vztahy mezi tělesy jako je průnik, překrytí, překřížení či vnoření. (viz. Obrázek 1)




Obrázek 1: Přehled vybraných interpretací vzájemných pozic (Dostupné z: [16])

## 1.4 Příklady geometrických jazyků

Následující kapitola uvádí příklady několika jazyků, které vznikly za účelem reprezentace geometrických útvarů. Většina těchto jazyků vychází z již známých jazyků, tyto původní jazyky však obohacuje o množství syntaxí a rozšíření, dalo by se tedy mluvit o jakémsi jazykovém rozšíření, či nadstavbě nad standardními jazyky.

### 1.4.1 GeoJSON

Standard GeoJSON vychází myšlenkově z formátu JSON, který je hojně užíván v jazyku Javascript. Hlavní rozšíření tohoto jazyka představuje přítomnost speciálních atributů, které reprezentují geometrickou informaci. Samotná geometrická informace je pak reprezentována prostřednictvím souřadnic. K dispozici jsou geometrie typu bod, typu křivka a také geometrie typu polygon. Formát jako takový je dobře čitelný a je možné jej přímo zkonvertovat pro použití v jazyku Javascript. Další výhodou tohoto formátu je malá velikost přenášených dat.

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

Výpis 1 Příklad formátu GeoJSON (dostupné z: [18] )

### 1.4.2 Well-known Text

Well-known text (WKT) je textový značkovací jazyk pro reprezentaci objektů vektorové geometrie. Díky tomuto jazyku je možné v textové podobě definovat prostorové objekty a také možné konverze

mezi jednotlivými projekčními formáty. Využití nachází převážně v prostorových referenčních systémech (Spatial Reference Systems – SRS), databázích a v zobrazování objektů na mapové podklady. Jeho binární ekvivalent Well-Known Binary (WKB) je používán k převodu a ukládání v databázových systémech jako jsou PostGIS, Microsoft SQL Server a DB2. [20] [21] [22] Je možné jej použít v prostorových referenčních systémech, pro zobrazování objektů na mapách, či k transformaci mezi prostorovými referenčními systémy (Spatial Reference Systems – SRS). Tyto formáty byly původně definovány konsorciem Open Geospatial Consorcium (OGC). Samotný standard [19] definuje strukturu a obsah tzv. Well-Known text řetězců. Nikterak však nepředepisuje, jak by měly implementace tento formát číst či zapisovat.

Oproti jazyku GeoJSON není účelem jazyka reprezentace dat vhodná pro přenos, dá se s ním však velice dobře pracovat v oblastech databází. Mnoho databázových programů podporuje tento formát jako vstupní formát pro definici geometrických objektů. Jak bylo zmíněno, je také velice dobrým nástrojem pro konverzi dat. Umožňuje reprezentaci souřadnic v jedné, dvou i třech dimenzích včetně možnosti použití 4. rozměru pro „lineární referenční systém“ (vzdálenost).

### 1.4.3 Simple Features

Jedná se o standard specifikující uložení objektů v databázi. Je spravován konsorciem „Open Geospatial Consorcium“ (OGC). Tento standard s tématem práce přímo nesouvisí, je zde však uveden jako exemplář formátu pro interní reprezentaci dat v databázi. Bližší informace o tomto standardu lze dohledat v [23] [24].

### 1.4.4 GML

Jazyk GML pak představuje standard, který je hodně využíván v oblastech geografických informačních systémů (GIS) ale také pro transakce s geografickými daty. [6] Disponuje rozsáhlými možnostmi pro reprezentaci geografických útvarů. Příkladem může být množství způsobů pro reprezenaci křivky, které jsou uvedeny v kapitole 2.4 Rozšiřující útvary detailnějšího popisu jazyka GML. Jeho výhodou je jak snadná strojová čitelnost, tak snadná čitelnost člověkem. Nevýhodou pak může být jeho neúspornost v porovnání s ostatními formáty.

### 1.4.5 AIXM

Posledním zmíněným standardem je pak AIXM (Aeronautical Information Exchange Model), který představuje standard pro výměnu leteckých dat. Jeho prostřednictvím mohou systémy spjaté s letectvím a leteckou dopravou provozovat výměnu dat. Poslední verze tohoto standardu (5.1) vychází z jazyka GML. [25]

# 2. Jazyk GML

Tato kapitola pojednává o jazyce GML. Samotný standard AIXM 5.1, který je součástí tématu této práce, přímo z tohoto jazyka vychází.

## 2.1 Účel jazyka

„GML je značkovací jazyk, který se používá k popisu geografických objektů ve světě okolo nás. Vzhledem k tomu, že je postaven na základech širších internetových standardů od World Wide Web Consorcium (W3C), jazyk GML vyjadřuje geografickou informaci ve formě, která může být snadno sdílena na internetu. “ [1] Jazyk GML jako takový staví na základech jazyka XML (eXtensible Markup Language). Díky tomuto může mít jazyk GML jednoznačně definovaný formát, podložený na základě jednoznačně definovaných schémat jazyka XML (XML schema).

Samotný jazyk bere inspiraci v realném světě. Reálné objekty jsou v něm nazývány jako tzv. Features. Každý takovýto objekt může být popsán pomocí množiny tzv. Properties – vlastností. Takovéto vlastnosti mohou představovat libovolné zpřesňující údaje o objektu, jako název, nosnost, adresu, ale hlavně geometrickou polohu, či reprezentaci objektu. Jako možný příklad takovýchto objektů může být prvek *KarlůvMost* odvozený od prvku Feature, představující reprezentaci Karlova mostu v Praze. Tento prvek může disponovat prvky odvozenými od prvku Property, jako například stáří mostu, počet mostních pilířů, datum další revize, či počet soch na ochozu. Důležitou a klíčovou vlastností pro jazyk GML je však vlastnost Property udávající geografickou pozici mostu. Pomocí této vlastnosti je možné velice věrně reprezentovat nejen přibližnou pozici mostu, ale také detailní tvar celého mostu, a to jak ve dvourozměrném pojetí, tak při nutnosti také celkové trojrozměrné reprezentaci celého mostu včetně detailů v podobně například již zmíněných soch.

Tento jazyk nachází uplatnění v geografických informačních systémech (GIS) a ve všech aplikacích, ve kterých je třeba přenášet data v nějaké standardizované formě.

## 2.2 Základní členění geometrií (knihovny)

Jazyk GML je strukturován do jednotlivých knihoven. Samostatnou kategorii tvoří knihovny geometrických prvků. Struktura těchto knihoven je převážně podle dimenzí objektů. Výjimku pak tvoří datové typy, které mají spíše agregační funkci. Pro účely této práce byla zvolena verze jazyka GML 3.2. Důvodem k výběru této verze byl fakt, že datový standard AIXM 5.1 je založen právě na knihovnách GML 3. Veškeré geometrie v GML 3 jsou obsaženy v následujících knihovnách [1]:

- geometrybasic0d1d.xsd
- geometrybasic2d.xsd
- geometryAggregates.xsd
- geometryPrimitives.xsd
- geometryComplex.xsd

## 2.3 Základní geometrické útvary

Základem jazyka GML jsou základní geometrické útvary z jazyka GML 2. Tyto geometrie se nacházejí ve schématech geometrybasic0d1d.xsd, geometrybasic2d.xsd. Obsahují základní geometrie pro reprezentaci bodu (Point), lomené čáry (LineString), uzavřené plochy (Polygon) a „uzavřené plochy s otvory“ (MultiPolygon). Blíže tyto prvky popisuje následující seznam:

- **Point** – bod reprezentovaný dvojicí souřadnic
- **LineString** – sekvence bodů spojených úsečkami
- **Polygon** – sekvence křivek tvořících plochu
- **MultiPolygon** – sekvence křivek tvořících vnější okraj a volitelné vnitřní křivky

## 2.4 Rozšiřující útvary

Všechny zmíněné geometrie se objevily již v jazyce GML 2, GML verze 3 však zavádí zcela nové možnosti reprezentace složitějších útvarů. Zavádí pojmy abstraktního datového typu Curve pro reprezenzaci křivek a datového typu Surface pro reprezentaci složitějších objektů. Veškeré objekty odvozené od objektu Curve jsou definovány v souboru geometryPrimitives.xsd. Objekt Curve může být složen z jednoho, či více křivkových segmentů:

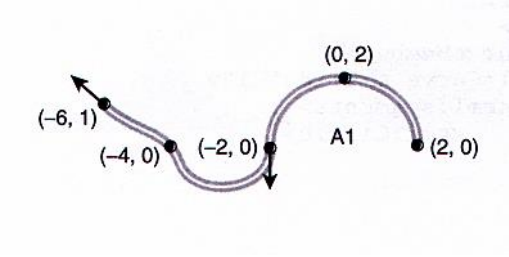
- **LineStringSegment** – Útvar tvořený množinou bodů, které jsou lineárně interpolovány v křivku. Tento datový typ je podobný typu LineString, avšak je substituovatelný za _CurveSegment, nikoliv za _Curve. Protože je LineString geometrický element z GML 2,

nejedná se o segment křivek a používá se pouze pro případy, které vyžadují základní geometrii. ([odkaz na 3 zdroj v svp] str.135)

- **Arc** – Část kružnice, která je definována pomocí tří bodů. Křivka je dopočítána na základě interpolace.
- **ArcString** – Křivka tvořena jednou, či více geometriemi typu Arc, spojenými vždy přes společný bod.
- **Circle** – Druh typu Arc, jehož první bod je identický s posledním
- **ArcStringByBulge** – Obdoba ArcString, s tím rozdílem, že zde odpadá definice „prostředních bodů“. Namísto nich se použije vzdálenost od přímky a normálový vektor.
- **ArcByBulge** – Křivka tvořená dvojící bodů představující úsečku, vzdáleností od ní a normálovým vektorem, určujícím směr od úsečky.
- **ArcByCenterPoint –** Druh typu Arc, který je definován souřadnicemi středu kružnice, dále počátečním (startovním) úhlem a koncovým úhlem.
- **CircleByCenterPoint** – Obdoba ArcByCenterPoint, kde startovní a koncový úhel musí být stejný.
- **CubicSpline** – Křivka parametrizovaná pomocí kubického splajnu. (definován jako počáteční vektor, koncový vektor a trojice koordinátů).
- **Bspline, BezierCurve** – Křivky parametrizované pomocí racionální funkce.

Jednotlivé subelementy prvku Curve jsou uloženy v jeho Property s názvem segments. Zajímavostí je rozmanitost možností reprezentace kruhových prvků (prvky s počátečním návem Arc). Takovýto prvek je zpravidla tvořen trojicí bodů, která značí počáteční bod, koncový bod a bod, kterým se má vést „půlkruh“. Jazyk GML však umožňuje datovému architektovi použití různých definic tohoto „prostředního bodu“. Je možné si vybrat z klasické reprezentace pomoci souřadnic (Arc), existuje však také reprezentace pomocí úhlového vektoru (ArcByCenterPoint) a reprezentace pomocí normálového vektoru od přímky tvořené počátečním a koncovým bodem (ArcByBulge). Samozřejmostí je také možnost vytvoření sekvence půlkruhů za pomocí množiny souřadnic. Takováto množina musí obsahovat informaci o počtu „půlkruhů“ (atribut numArc).

Možný příklad vizualizace datového typu Curve ukazuje Obrázek 2, který obsahuje segmenty Arc a CubicSpline.




**Obrázek 2: Příklad křivky Curve složené z typů Arc a CubicSpline (dostupné z: [1])**

Objekt Surface obsahuje veškeré subelementy v Property s názvem patches. Na rozdíl od prvku Curve obsahuje Property patches jeden a více tzv. suface patches, které dohromady tvoří výsledný povrch. Typickým případem takového surface patch je prvek PolygonPatch (nikoli Polygon). [1] Tento prvek se skládá z povinné vnější hranice (element exterior) a nula či vice vnitřních hranic (prvek interior)

Až doposud byly veškeré datové typy tvořeny jednou či více spojenými geometriemi. Knihovna geometryAggregates.xsd však obsahuje takové datové typy, které mohou obsahovat i libovolný počet geometrických objektů, které na sobě nemusí záviset. Takovéto typy geometrií plní funkci agregace

jednotlivých pod geometrií do celku (odtud také název této knihovny). Z příkladů těchto geometrií je možné uvést následující:

- **MultiPoint** – kolekce jedné a více instancí datového typu Point
- **MultiCurve** – kolekce jedné a více instancí datového typu Curve
- **MultiSurface** – kolekce jedné a více instancí datového typu Surface

Zbývající knihovna geometryComplex.xsd obsahuje objekty, díky nimž je možné sdružovat veškeré geometrie do nových konkrétních geometrických objektů. Pod takovými objekty si lze představit například konstrukci objektu reprezentující silniční síť (viz. [GML kniha] str.148).

- **CompositeCurve** – Křivka složena z geometrických typů *Curve.*
- **CompositeSurface** – Povrch složený z geometrických typů *Surface*
- **CompositeSolid** – Objekt složený z geometrických typů *Solid*
- **GeometryComplex** – Objekt složený z veškerých geometrií

Výše uvedené seznamy geometrií byly zmapovány v rámci výzkumného projektu [8]

## 2.5 Koncept Feature a Property

Jak již bylo uvedeno na počátku této kapitoly, jazyk GML přichází s koncepcí objektu Feature a jeho vlastnostmi Property. Objekt Feature reprezentuje jakýkoli zamýšlený objekt. Jednoznačnou identifikaci objektu zajišťuje identifikátor gml:id. Tento identifikátor je přítomen v elementu daného feature v podobě atributu a měl by být unikatní v rámci celého XML dokumentu. Dalším prvkem objektu Feature je výčet jeho vlastností Property, které jej zpřesňují. Objekty Property nemusí být přítomny pouze lokálně, existuje také možnost přidání objektů Property ve formě odkazů. Takovéto odkazování se provádí pomocí technologie xLinks za použití odkazu na identifikátor daného prvku (identifikátor gml:id).[1][6] Více o této technologii je napsáno v kapitole Reference v AIXM 5.1, který tuto technologii odkazování přejímá.

## 2.6 Projekční systémy

V jazyce GML je projekční systém vyjadřován u jednotlivých geometrií za pomocí atributu srsName. Tento atribut obsahuje informace k identifikaci použitého projekčního systému. V praxi je možné se setkat s reprezentací pomocí tzv. Unified Resource Locator. Příklad tohoto formátu ukazuje následující příklad atributu deklarující použití projekčního systému WGS84. Jako hodnotu je možné do atributu srsName uložit také odkazy xLink.

```
srsName="urn:ogc:def:crs:EPSG:4326"
```

**Výpis 2: Příklad deklarace atributu srsName**

Dědičnost souřadného systému pokračuje do celé hloubky elementu. Jeho platnost je přepsána pouze v případě výskytu dalšího, lokálního atributu srsName. Od toho okamžiku všichni potomci tohoto elementu přebírají jeho projekční systém. [26]

## 2.7 Zápis geometrických souřadnic

Zápis geometrických souřadnic je možné v jazyce GML uskutečnit pomocí jedním z níže uvedených elementů. Z důvodu umožnění použití různých souřadnicových systémů GML disponuje atributem srsName. Tento atribut jednoznačně definuje, v jakém souřadnicovém systému (CRS – Coordinate

Reference Systém) jsou data v elementu zapsána. Při zápisu geometrických souřadnic u prvků je možné zvolit z následujících elementů:

Element **pos** – představuje zápis jedné geometrické souřadnice. Jednotlivé koordináty jsou zapsány vedle sebe a odděleny pomocí mezer. „Do vlastnosti pos mohou být uvedeny také atributy dimension a srsName“[1]. V případě potřeby deklarace více souřadnic je nutné uvést několik těchto elementu za sebou.

Element **posList** – tato reprezentace je tvořena seznamem všech souřadnic odděleným mezerami. Pro lepší parsování tohoto pole je v záhlaví elementu uveden atribut dimension specifikující dimenzi výsledného seznamu souřadnic.

Element **coordinates** – reprezentuje geometrické souřadnice ve formátu oddělujícím jednotlivé složky souřadnice pomocí čárky. Celé „vektory“ souřadnic jsou pak odděleny za pomoci mezer. Tento element bohužel neumožňuje definici souřadnicového systému (atribut srsName), a tudíž je nutné tuto informaci uvést v některém z předků tohoto elementu.

Nutno poznamenat, že ne každý geometrický prvek podporuje všechny zmíněné formáty souřadnic. Dále je třeba zmínit, že výsledný formát souřadnic resp. pořadí jednotlivých dimensionálních složek je podřízeno zvolenému projekčnímu systému (CRS).

# 3. AIXM 5.1.

Tato kapitola pojednává o standardu pro výměnu dat AIXM 5.1. Tento standard přímo vychází z jazyka GML, doplňuje jej však o nové vlastnosti.

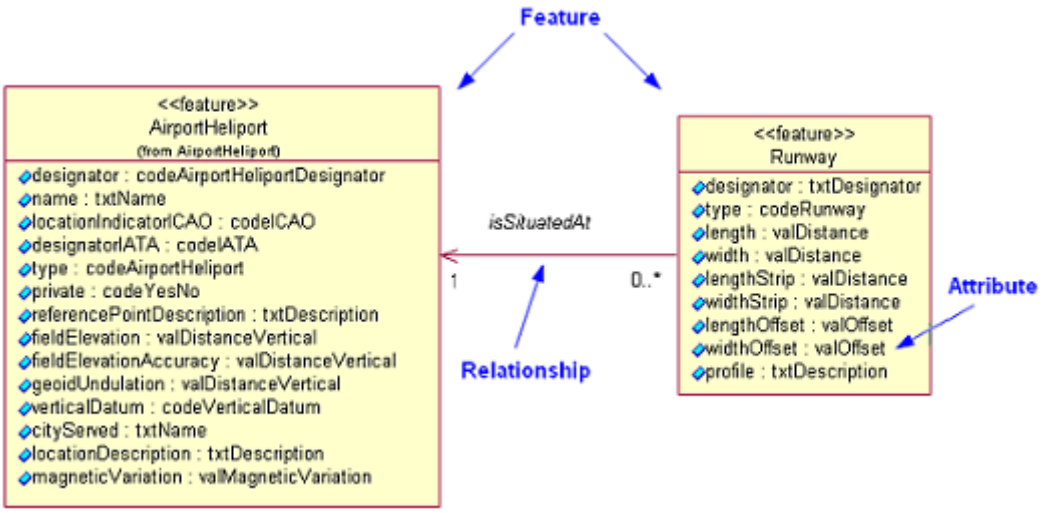
## 3.1 Základní koncepce jazyka

AIXM je společným projektem organizací FAA a EUROCONTROL pro práci s leteckými daty. „Aeronautical Information Exchange Model (AIXM) je navržen tak, aby umožnil správu a distribuci dat Aeronautical Information Services (AIS) v digitální podobě.“ [27] AIXM 5.1 pak představuje novou vývojovou verzi, která nahrazuje starší verzi AIXM 4, avšak zásadně mění celkovou koncepci dat a přidává několik nových funkcionalit.

Jako takový, je protokol AIXM tvořen několika subsystémy [28]:

- AIXM Konceptuální model
- AIXM XML schéma

Konceptuální model je založen na jazyku UML. Tento model detailně popisuje letecká data. Jak zobrazuje Obrázek 3, tento model se skládá z objektů (tzv. „Features“), jejich atributů (tzv. „Attributes“) a vztahů mezi objekty (tzv. „Relationships“).




Obrázek 3: Příklad konceptuálního modelu (dostupné z [27])

XML schéma pak slouží jako model pro výměnu dat mezi aplikacemi. Jeho funkcí je reprezentace konceptuálního modelu ve strojově čitelné podobě. Jako takové vychází XML Schema ze schématu jazyka GML.

## 3.2 Prvek Feature

Prvek Feature se dá považovat za jakýsi základní prvek ve struktuře AIXM dat. Jeho účelem je reprezentace aeronautických objektů. Jedná se o odvozený typ od typu Feature z schématu jazyka GML. Jeho účelem je, obdobně jako v případě jeho předka, reprezentace libovolného reálného či symbolického objektu na Zemi.

Prvek Feature obsahuje jednoznačný identifikátor. „S odkazem na koncept Temporality standardu AIXM, je prvek typu Property – identifier jedinou časově neměnnou vlastností; z tohoto důvodu je situován vně prvku TimeSlice, který zaobaluje veškeré prvky Property, které se mohou s časem měnit.“ O temporálním modelu pojednává následující kapitola. [32]

Dalším možnou identifikací prvku Feature je identifikace pomocí identifikátoru gml:id. Tento identifikátor, který je převzat z jazyka GML jednoznačně charakterizuje prvek feature, avšak pouze v rámci daného XML dokumentu. Je tak možné jej použít pro odkazování se na Feature do dokumentu a také pro případnou kontrolu přítomnosti všech žádaných prvků v dokumentu.

V praxi se můžeme setkat například s prvky Feature reprezentujícími letiště/heliport (prvek AirportHeliport) či Airspace reprezentující letovou oblast [29] [30]. Kompletní výčet všech prvků Feature lze nalézt v oficiálním schématu standardu AIXM 5.1 (soubor Features.xsd) dostupném v [27].

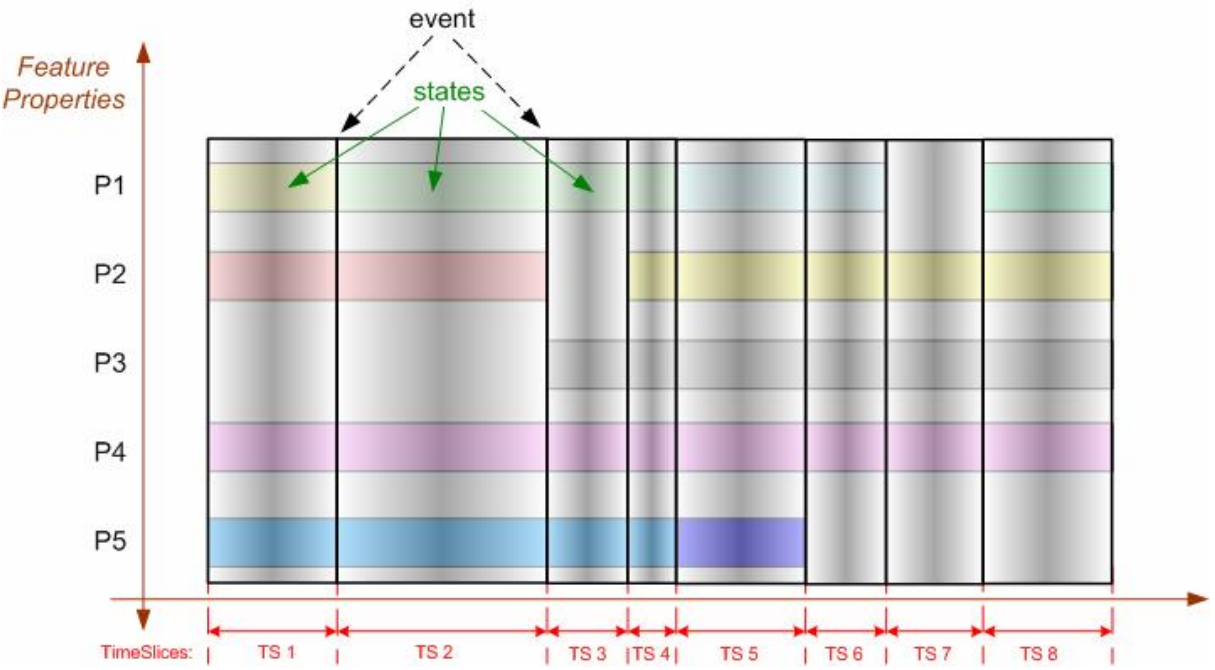
## 3.3 Temporální model

Aeronautická data nejsou v průběhu času stálá. Mohou se v čase měnit, zanikat, či vznikat nová. Vzhledem k tomuto faktu je třeba nějakým způsobem uchovávat historii aeronautických dat. Standard AIXM 5.1 přichází s konceptem tzv. „Temporálního modelu“ (Temporality model). „Základní temporální model by měl být uniformně aplikovatelný na všechny aeronautické typy prvků a koncept temporality by měl být nezávislý na návrhu vlastností jednotlivých prvků.“ [31]

Jak již bylo řešeno dříve, data AIXM lze rozdělit na Features, Attributes a Relationships. Základním stavebním kamenem všech dat je Feature, který může obsahovat Attribute a Relationship (v oficiálních

materiálech [27] souhrnně nazýváno jako Property). Aby bylo docíleno jednoznačnosti dat v čase, standard zavádí pojem *Stavu* a *Události* (State a Event).

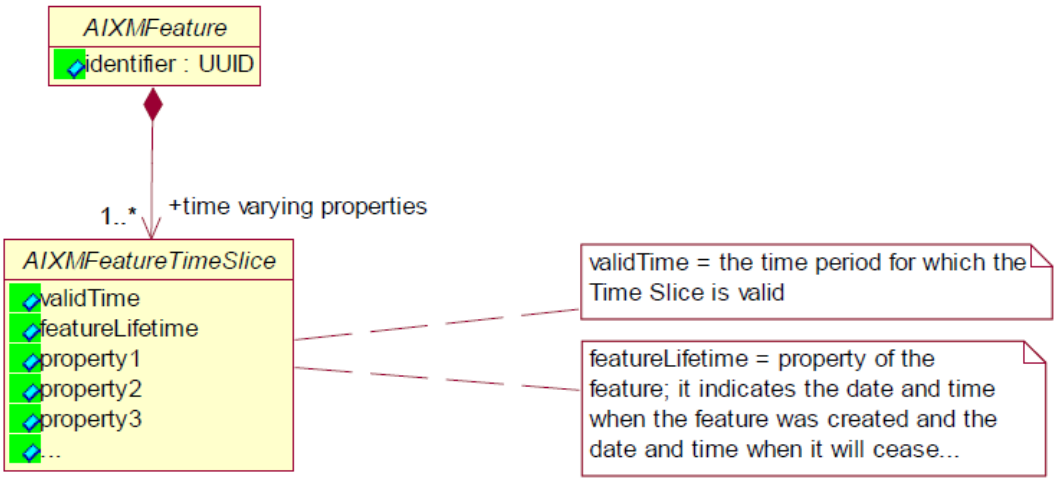
*Stav* (State) představuje atomickou a neměnnou entitu, která reprezentuje aktuální stav Property nějakého objektu, v daném období. *Událost* (Event) pak představuje změnu, mezi dvěma stavy. Takováto *událost* může představovat změnu jedné, či více Property dané Feature. Na Obrázek 4 je zobrazen graf libovolné Feature, která obsahuje Property P1 až P5 (v obrázku zobrazeny na ose Y) na ose X pak figurují jednotlivé *stavy* a *události.*




**Obrázek 4: Přehled vývoje jednotlivých Property v čase (Dostupné z [31])**

Takovéto *události* a s*tavy* je však třeba nějak vhodně reprezentovat. Z tohoto důvodu obsahuje každá Feature kolekci tzv. TimeSlices. TimeSlice představuje konkrétní reprezentaci *Stavů* jednotlivých Properties dané Feature, přesněji obsahuje aktuální stav všech Property v daném časovém období. Podíváme-li se znovu na Obrázek 4, jednotlivé soubory *stavů* mezi dvěma *událostmi* tvoři TimeSlice (na obrázku znázorněny jako interval na ose X s popiskem TS<číslo>) u*dálosti* pak představují hranice mezi jednotlivými TimeSlices.

Hierarchie prvků v AIXM 5.1 pak vkládá prvek TimeSlice jako jakýsi „kontejner“ pro Property. (viz. obrázek)




Obrázek 5: Vztah AIXMFeature a AIXMTimeSlice (Dostupné z [31])

### 3.3.1 Trvalé a dočasné změny

V reálném provozu nemusí docházet pouze k trvalým změnám. „Aeronautické prvky mohou být ovlivněny dočasnými událostmi, jako například uvedení navigačního zařízení mimo provoz, zavření přistávací dráhy, zpřístupnění zakázané letové oblasti. Všechny tyto události vytvářejí dočasné změny, v hodnotách jedné, či více Property daných Feature. Po ukončení platnosti těchto událostí se hodnoty těchto Property vrátí na původní hodnotu“ [31]

Standard AIXM 5.1 tento problém řeší zavedením dočasných a trvalých prvků Timeslice. Aby bylo možné jednoznačně identifikovat o jaký druh TimeSlice se jedná, obsahuje prvek TimeSlice povinnou položku identification.

Přípustné hodnoty jsou:

- BASELINE – trvalý stav Feature (přehled všech hodnot)
- PERM_DELTA – výčet všech trvale změněných hodnot
- TEMP_DELTA – dočasný stav Feature (výčet všech dočasně změněných hodnot)
- SNAPSHOT – aktuální stav Feature (průřez aktuálními hodnotami všech Property v daném čase)

#### *3.3.1.1 BASELINE*

Tento typ prvku obsahuje aktuální obraz všech Property daného Feature. „Baseline Time Slice obsahuje hodnoty všech Properties, jež byly definovány od času platnosti TimeSlice“[31]

#### *3.3.1.2 PERM_DELTA*

„Jedná se o druh TimeSlice, který popisuje rozdíl mezi jednotlivými stavy Feature v důsledku trvalé změny“[31]. Tento typ prvku byl zaveden z důvodu potřeby určitých druhů systémů, (v [31] uváděných jako Push systémy). Takovéto systémy, z důvodu úspory dat, nevyžadují celkový přehled všech aktuálních Properties, nýbrž vystačí s pouze s informací, které konkrétní hodnoty (Property) byly změněny. Na Obrázek 4 je možné si představit jako *události* (Events).

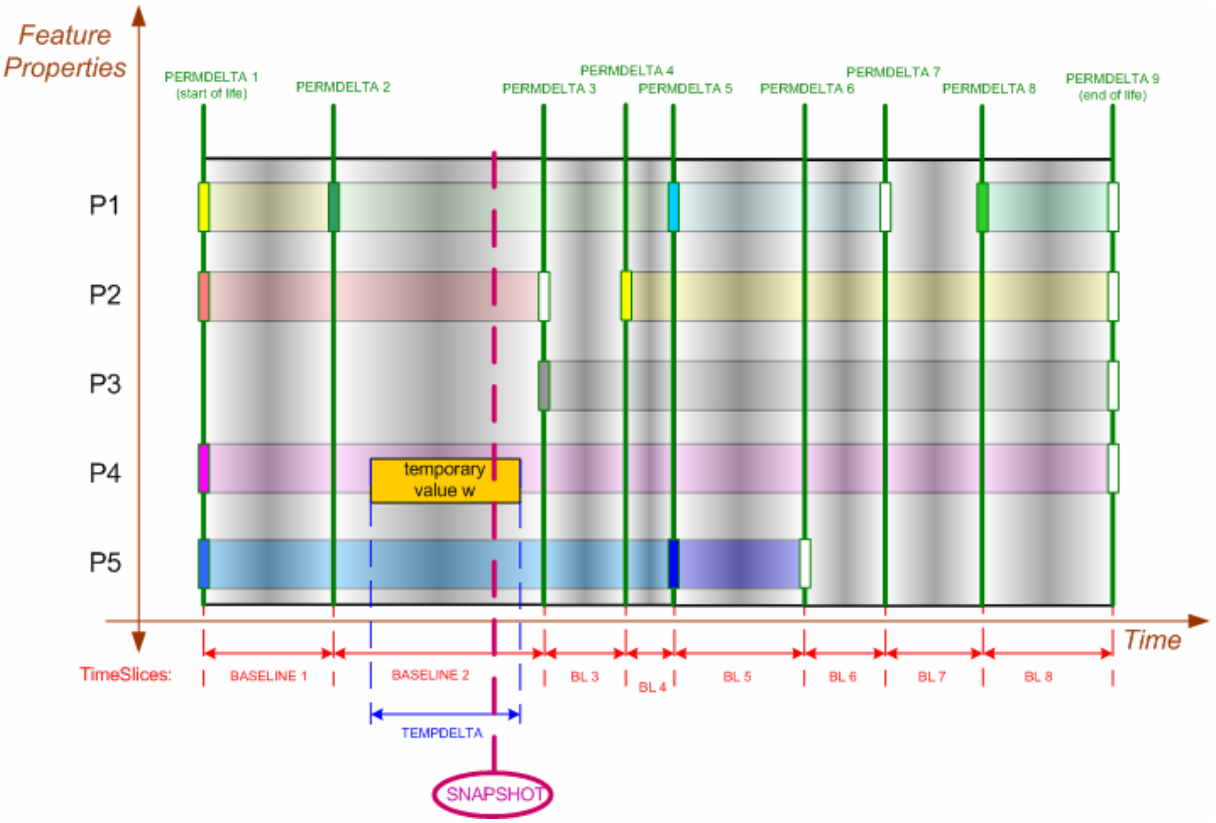
#### *3.3.1.3 TEMP_DELTA*

„Druh TimeSlice, který popisuje přechodné překrytí stavu během dočasné události“[31]

#### *3.3.1.4 SNAPSHOT*

„Jedná se o druh TimeSlice, který popisuje stav Feature v daný čas, coby výsledek kombinace současného BASELINE TimeSlice platného v daném čase se všemi TEMPDELTA TimeSlice, které jsou platné v daný čas.“ Díky tomuto typu TimeSlice je možné zobrazit aktuální stav Feature, včetně aplikovaných dočasných změn.

Závěrem této kapitoly je předložen obrázek, který graficky znázorňuje veškeré typy prvku TimeSlice




Obrázek 6: Vývoj prvků Property v rámci prvků TimeSlice (Dostupné z: [31])

### 3.3.2 Verzování atributů Feature

V případě několika změn v čase se vytvoří příslušný počet prvků TimeSlice. Jednotlivé prvky je však nutné mezi sebou rozeznat. K tomuto účelu slouží povinný atribut SequenceNumber. Jedná se o číselný typ, jež značí, o kolikátou verzi úprav v čase se jedná. Atribut sequence number hraje roli unikátního identifikátoru pro každý prvek TimeSlice ve Feature. [31] Nutno poznamenat, že atribut Sequence number zajišťuje unikátnost pouze v rámci časových úprav (nikoli zpětných korekcí hodnot). Komplexní unikátnost zaručuje kombinace atributů SequenceNumber a CorrectionNumber, který je představen v následující kapitole.

### 3.3.3 Korekce, změny hodnot u Feature

V případě potřeby správy změn v průběhu času standard AIXM disponuje atributem SequenceNumber. Problém však nastane v případě, kdy je třeba provést zpětnou korekci hodnot. V takovémto případě může dojít k nekonzistenci dat. Z tohoto důvodu prvek TimeSlice obsahuje povinný aribut CorrectionNumber, který eviduje verzi prvku v rámci zpětných korekcí.

V kombinaci s atributem SequenceNumber tak atribut Correction Number jednoznačně identifikuje každý TimeSlice. Jedinou výjimku tvoří TimeSlice typu SNAPSHOT, kde jsou tyto atributy záměrně vynechány.

## 3.4 Reference v AIXM 5.1

Z důvodu úspory dat umožňuje standard AIXM 5.1 možnost odkazovat se na jednotlivé prvky. Takováto možnost se hodí v případě, kdy je třeba definovat hodnotu Property, avšak samotná hodnota je dosti velká či složitě strukturovaná, případně chceme zamezit zbytečnému přenosu dat. (Nutno poznamenat, že v AIXM 5.1 je možné odkazovat se pouze na prvky **Feature**. Ostatní objekty Property musí být uvedeny v **celé šíři**)

AIXM 5.1 řeší reference za použití jazyka XLink. Je možné vybrat si z následujících typů referencí [32]:

- Konkrétní reference v rámci datové sady
- Konkrétní externí reference řešené pomocí webových služeb
- Abstraktní reference, které mají být vyřešeny externí aplikací

### 3.4.1 Konkrétní reference v rámci datové sady

V některých případech, služby produkující data ve formátu AIXM 5.1, poskytují datové sady, které rovněž obsahují všechny odkazované prvky Feature. Kromě odkazování se za pomoci hodnoty atributu gml:identifier, je možné použít také lokální odkaz na přes gml:id atribut. Gml:id atribut je dle definice unikátní v rámci xml dokumentu. Proto je při použití v rámci lokálního odkazování zcela jednoznačný. Atributy gml:id jsou ve schématu uvedeny jako identifikátory a mohou být během parsování indexovány. [32]

```
xlink:href="#uuid.a82b3fc9-4aa4-4e67-8defaaea1ac595j"
```

Výpis 3: Příklad lokální reference v AIMX 5.1

### 3.4.2 Konkrétní externí reference řešené pomocí webových služeb

V případě, kdy jsou informace o prvcích Feature vystaveny v rámci webových služeb, je způsob přístupu k referencím přes XLinks tvořen přes Universal Resource Locator (URL) adresu. „V takovémto případě se očekává, že příjemce zprávy může následovat URL podporovanou adresu XLink přímo a hashtag pak jednoznačně identifikuje prvek Feature v rámci výsledného zdroje, kam reference ukazuje.” [32]

```
xlink:href="http://aim.faa.gov/services/AirspaceService#uuid.a82b3fc
9-4aa4-4e67-8def-aaea1ac595j"
```

Výpis 4: Příklad externí reference v AIXM 5.1

### 3.4.3 Abstraktní reference, které mají být vyřešeny externí aplikací

„XLink předpokládá použití „resource centred“ přístupu, ve kterém je XML dokument, či fragment dokumentu zdroj (resource), na který je možné se odkázat odkudkoli. Předpokládá se, že zdroj je dostupný na webu v jedné definitivní kopii. V aeronautické informační doméně je prvek Feature entitou, na kterou je možné se globálně odkazovat, avšak v rámci všech systémů existuje množství reprezentací v pohybu. Většinu z těchto reprezentací tvoří AIXM zprávy, ostatní reprezentace jsou uloženy v databázích či aplikacích. Z tohoto důvodu je použití konkrétní reference spíše vzácností a je třeba počítat také s abstraktní referencí.“ [32]

```
xlink:href="urn:uuid:39db4c6c-32c5-4770-b2d2-7e97af0d47cf"
```

Výpis 5: Příklad abstraktní reference v AIXM 5.1

### 3.4.4 Postup při řešení reference

„Důležitou zmínkou při použití abstraktní reference je fakt, že nic při jejím použití nezaručuje dostupnost referencovaného prvku Feature, či dostupnost jeho umístění; může nastat situace, kdy je prvek Feature definován lokálně v rámci příchozí zprávy, dostupný vzdáleně prostřednictvím webové služby, či přímo z databáze.“ [32]

Při hledání referencovaného prvku obecně platí, následování těchto třech kroků:

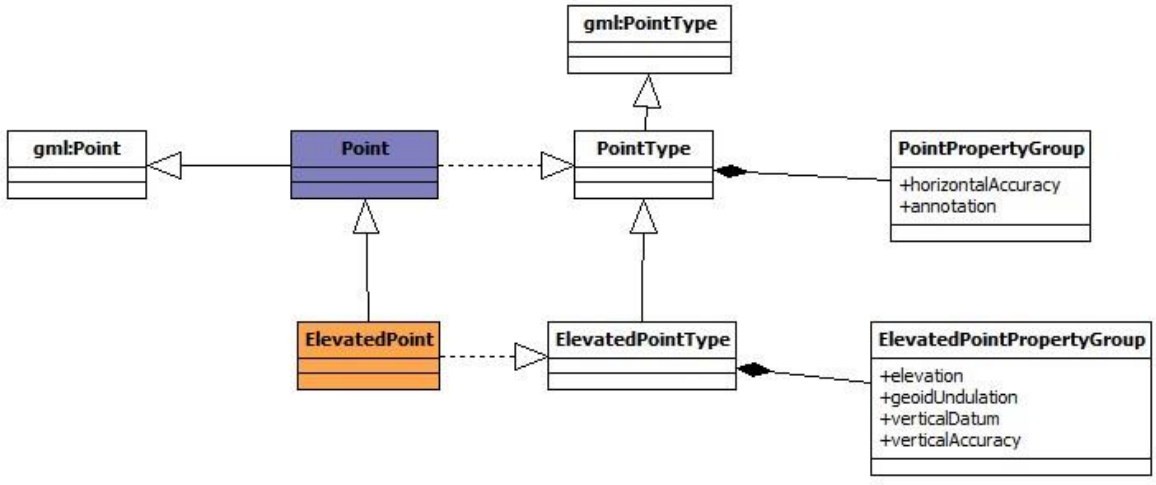
1) Příjemce použije identifikátor referencovaného prvku Feature k prohledání své lokální databáze
2) Pokud takový prvek neexistuje v lokální datové sadě, dojde k prohledání příchozí datové sady na přítomnost referencovaného prvku Feature
3) V případě nenalezení prvku Feature, by měl systém prohledat známé datové zdroje na přítomnost referencovaného prvku Feature

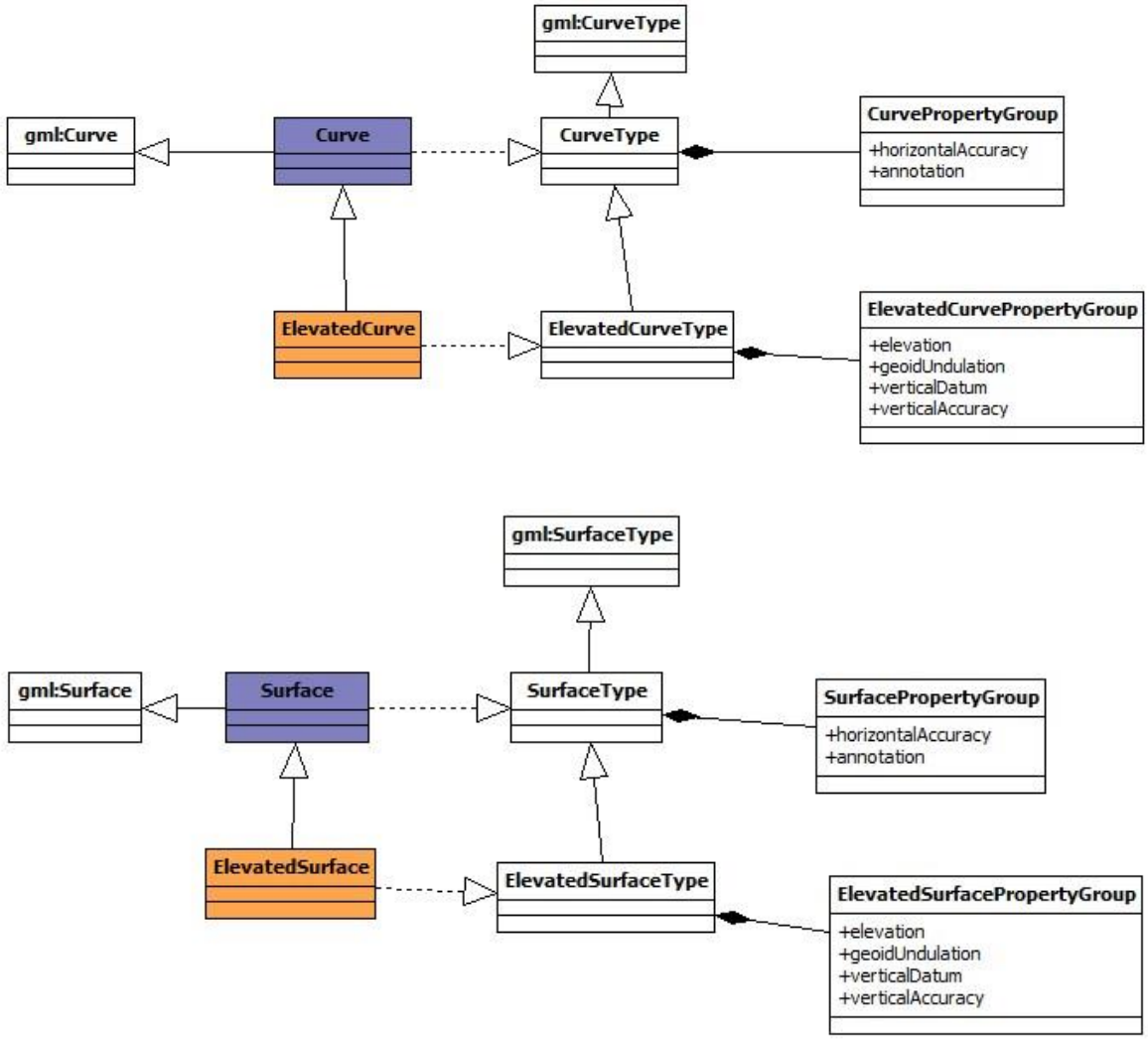
## 3.5 Prvek Object

Tento prvek představuje entitu, která sama o sobě nemá statut prvku Feature. Může se však vyskytovat v rámci jednotlivých Property u prvku TimeSlice. Za jeden z exemplářů těchto prvků je možné označit například prvky Property obsahující AIXM geometrie.

## 3.6 Systém geometrií v AIXM 5.1

Geometrie ve standardu AIXM 5.1 jsou odvozeny ze základních geometrií v jazyce GML. Při studiu aixm schémat byl vytvořen následující diagram:






Obrázek 7: Diagram struktury geometrií v AIXM 5.1

Z diagramu je patrné, že se v modelu vyskytují tři hlavní kategorie. Jsou to *aixm:Point*, *aixm:Curve* a *aixm:Surface* (v diagramu označeny **modře**). Tyto geometrie jsou odvozeny od svých jmenovců z jazyka GML. Každá z geometrií však přidává dvě nové položky: *horizontalAccuracy* a *annotation*.

Od každého z výše zmíněných typů pak dědí třída s předponou „*Elevated*“, která obsahuje odlišnou sadu atributů. Jedná se o atributy *Elevation, geoIdUndulation, verticalDatum* a *verticalAccuracy*. Jednotlivé datové typy jsou v diagramu označeny **oranžově**.

## 3.7 AIXM 5.1 dokument

Klasická AIXM zpráva je tvořena hlavičkou s deklaracemi použitých hlaviček ve zprávě. Následuje sekvence párových elementů *hasMember*, která představuje zapouzdření pro jednotlivé části zprávy. V každé části zprávy pak figuruje prvek *Feature*, který obsahuje nula až nekonečno prvků *TimeSlice*. Prvek *TimeSlice* pak obsahuje jednotlivé elementy s prvky *Property*.

Jak bylo popsáno v předchozích kapitolách, pod prvkem *Property* je možné představit si konkrétní hodnotu (*Attribute*), prvek typu *Object*, či prvek reference (*Relationship*) . V případě konkrétní hodnoty je tato hodnota uvedena přímo do elementu. V případě prvku typu *Object* je však nutné uvést v elementu celou reprezentaci včetně všech vlastností. Prvky *Feature* se v rámci prvků *Property*

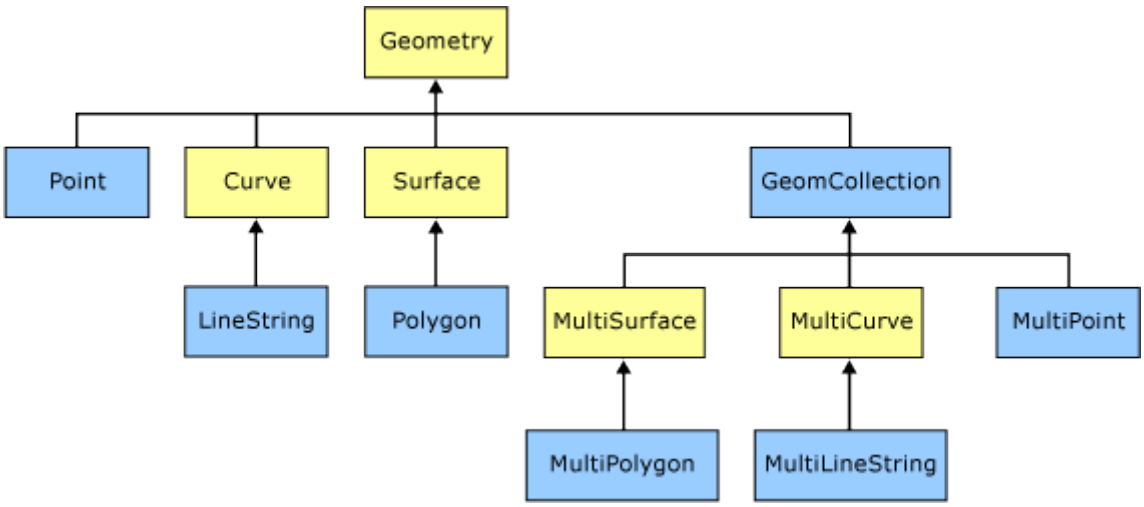
vyskytují pouze jako reference. V příloze této práce, v adresáři `/testData` je uveden příklad takovéhoto dokumentu. Jedná se o dokument, který obsahuje jeden vzorový prvek *Feature*.

# 4. Srovnání geometrií se systémem MS SQL Server Spatial

Následující kapitola popisuje přehled systému geometrií v Microsoft SQL Server Spatial. Pro účely práce byla vybrána verze Microsoft SQL Server Spatial 2014. Jednotlivé geometrie jsou následně srovnány s geometriemi GML. Z tohoto srovnání následně vychází specifikace implementace.

## 4.1 Geometrie v Microsoft SQL server spatial

Celkový koncept geometrií v Microsoft SQL Server Spatial je založen na dvou odlišných reprezentacích geografických informací. „SQL Server podporuje dva typy geometrických dat: datový typ geometry a datový typ geography.“ [8] Rozdíl v těchto typech je ve způsobu reprezentace geografických objektů. Datový typ geometry reprezentuje data v Eukleidovském (plochém) souřadném systému, kdežto datový typ geography reprezentuje data ve sférickém souřadnicovém systému. Při použití datového typu geography je tak potřeba definovat v jakém souřadnicovém systému se data objektu nachází. Následující Obrázek 8 vyobrazuje hierarchii geometrií, od nichž jsou odvozeny datové typy geometry a geography. Instantiovatelné datové typy od typů geometry a geography jsou označeny modrou barvou




Obrázek 8: Přehled geometrií v Microsoft SQL Server (dostupné z [33]

Žlutě jsou vyznačeny abstraktní datové typy, z nichž jednotlivé datové typy vychází. Nutno poznamenat, že na Obrázek 8 jsou vyznačeny pouze základní geometrie. Z tohoto důvodu následuje popis jednotlivých instanciovatelných geometrií:

- **Point** – bod reprezentován pomocí souřadnic (latitude, longtitude)
- **LineString** – sekvence bodů spojená pomocí úseček
- **CircularString** – sekvence křivek, která je definována pomocí trojic bodů
- **CompoundCurve** – křivka složená z typů LineString a CircularString
- **Polygon** – plocha, která je definovaná jedním vnějším útvarem (množinou bodů) a nula či více vnitřními útvary (množinami bodů)
- **CurvePolygon** – obdobná struktura jako Polygon, avšak útvary mohou být tvořeny křivkami (CircularString, CompoundCurve)
- **MultiPoint** – kolekce nula či více bodů

- **MultiLineString** – kolekce nula či více LineString instancí
- **MultiPolygon** – kolekce nula či více Polygon instancí
- **GeometryCollection** – kolekce nula či více Geometry/Geography instancí

## 4.2 Podobnosti datových struktur s GML 3.2

Již z obrázku (Obrázek 8) je na první pohled patrné, že se v něm vyskytují datové typy nápadně podobné geometriím z jazyka GML. Pro příklad možno uvést jednoduchý datový typ *Point*, který je de facto se svým protějškem z GML identický. Jediným rozdílem v tomto datovém typu je striktní pořadí souřadnic, které je v případě datového typu *Point* v GML podřízeno zvolenému projekčnímu systému. Dalším takovým případem je datový typ LineString. Tento datový typ je takřka identický s datovým typem *LineString* v jazyce GML. Další podobnost můžeme nalézt také mezi datovým typem SQL Serveru Polygon a datovými typy z GML *PolygonPatch* a *Polygon*. V tomto případě se datový typ SQL Serveru jeví jako univerzálnější, neb umožňuje uložit jak uzavřený útvar, definující hranici plochy, tak případné vnitřní „otvory" v ní.

Pokud se podíváme na geometrie v SQL Server Spatial z pohledu knihoven GML, zjistíme, že geometrie *Point, LineString, Polygon* a *MultiPolygon* jsou schopné obstojně namapovat své protějšky z GML knihoven **geometrybasic0d1d.xsd** a **geometrybasic2d.xsd**.

Datový typ *CircularString* je dosti podobný datovému typu z GML *Arc* a jeho mutacím (*ArcStringByBulge, ArcByBulge* atd.). Stejně jako v případě GML typů se datový typ *CircularString* skládá z bodů, které definují, kudy má procházet křivka odvozená od části kružnice. Je tedy patrné, že případný převod by mohl být jednoduchý. Jedinou vadou zůstává množství možných reprezentací jednotlivých mutací prvku *Arc*. V takovémto případě by se muselo přistoupit ke konverzi do nějakého výchozího datového typu - nejlépe do datového typu *Arc*.

Datový typ *CompoundCurve* je možné srovnat hned s několika geometriemi z GML. Vzhledem k možnosti datového typu uložit jak geometrie typu *LineString* tak *CircularString*, je možné nalézt podobnosti mezi více datovými typy. V případě přítomnosti pouze prvků *LineString* je možno srovnat tento datový typ s datovým typem *LineStringSegment* v GML. V případě pouhé přítomnosti prvků *CircularString* je možné najít podobnosti s datovým typem v GML – *ArcString*. Pokud datový typ *CompoundCurve* obsahuje kombinaci obou datových typů, je možné srovnat tento datový typ s abstraktním datovým typem z jazyka GML – *Curve*. Z jistého pohledu je možné také srovnat datový typ *CompoundCurve* s datovým typem *MultiCurve*. Tento datový typ však umožnuje uložení i takových geometrií typu křivka, které na sebe nemusí navazovat. Z tohoto důvodu se podobnost těchto datových struktur, oproti výše zmíněným, jeví jako dosti volná.

Datový typ *CurvePolygon* také přímou alternativu v jazyce GML nemá. Pokud se však na něj podíváme z pohledu geometrií, které lze do něj uložit, zjistíme, že umožňuje uložit podobné geometrie jako tomu je v případě datového typu *CompoundCurve*. Tento typ však nesdružuje své geometrie do křivky, nýbrž do plochy. Jednou z možností je tedy podobnost s GML datovou strukturou *Polygon* – v případě použití linerárních geometrií. Možným dalším datovým typem jest *ArcString*, tento typ však nereprezentuje plochu, avšak pouze křivku. Z tohoto důvodu se jedná spíše o vzdálenou podobnost.

Datový typ *MultiPoint*, co se do podobnosti týče, disponuje takřka identickým protějškem v jazyce GML. Jeho protějšek má rovněž název *MultiPoint*.

Pro datový typ *MultiLineString* nedisponuje jazyk GML jednoznačný ekvivalentem. V rámci výzkumného projektu [8] však byla vypozorována jistá podobnost s datovými typy *LineStringSegment* a *MultiCurve*. V případě datového typu *LineStringSegment* je tato podobnost pouze částečná. Tento

datový typ totiž umožňuje uložit pouze spojitou sekvenci úseček. Naproti tomu datový typ *MutliCurve* sice již dokáže uložit i nespojité lomené čáry, dokáže však také uložit libovolné křivkové útvary odvozené z GML datového typu *Curve*. Z tohoto hlediska se tedy jeví jako naddimenzovaný.

Posledním datovým typem k porovnání je datový typ *GeometryCollection*. Tento typ se nápadně podobá datovému typu GML – *GeometryComplex*. Vzhledem k funkci obou datových typů, jež je zastřešení všech ostatních datových typů, mohou nastat situace, kdy se v jedné datové instanci budou vyskytovat datové typy, které zde byly popsány jako ne zcela jednoznačně ekvivalentní. Z tohoto důvodu nelze prohlásit tyto typy za zcela identické.

Závěrem této kapitoly je předložena tabulka odhalených podobností jednotlivých geometrií. Jednotlivé řádky tabulky obsahují vždy datový typ SQL Serveru Spatial a na druhé straně jemu podobnou strukturu z jazyka GML. V případě podobnosti dvou struktur, hraničící s identitou je uvedena značka ⇔, značící vzájemnou substituovatelnost. V případě pouhé podobnosti mezi datovými strukturami, avšak bez triviální konverze, je mezi datovými typy uveden znak '?'.

**Tabulka 1: Přehled podobností datových typů SQL Serveru Spatial a geometrií GML**

| SQL server spatial | | GML |
|---|---|---|
| Point | ⇔ | Point |
| LineString | ⇔ | LineString |
| CircularString | ⇔ | Arc, Circle, ArcBy*,CircleBy* |
| CompoundCurve | ? | MultiCurve, Arc, LineStringSegment |
| Polygon | ⇔ | Polygon |
| CurvePolygon | ? | Polygon, ArcString |
| MultiPoint | ⇔ | MultiPoint |
| MultiLineString | ? | LineStringSegment, MultiCurve |
| MultiPolygon | ⇔ | MultiPolygon |
| GeometryCollection | ? | GeometryComplex |

## 4.3 Mapování geometrií mezi GML a SQL Server Spatial

V rámci výzkumného projektu byl zjištěn fakt, že většina geometrií má podobnou myšlenku reprezentace a tudíž je převeditelná. „Ze zjištěných podobností v tabulce (Tabulka 1) vyplývá, že z SQL je drtivá většina struktur do GML převeditelná. Problém však nastává v opačném převodu, kdy GML disponuje specifickými strukturami." [8] Jedná se zejména o různé možnosti reprezentací stejných typů útvarů (datové typy GML odvozené od typu Arc).

### 4.3.1 Mapování geometrií z SQL Server Spatial do GML

Jak již bylo uvedeno v kapitole 4.2 Podobnosti datových struktur s GML 3.2, při zběhlém pohledu lze dojít k faktu, že existuje část geometrií v SQL Server Spatial, která je v podstatě identická se svými protějšky v GML. Jedná se o následující geometrie:

- *Point => Point*
- *LineString => LineString*
- *Polygon => Polygon*
- *MultiPoint => MultiPoint*
- *MultiPolygon => MultiPolygon*

Případné konverze mezi těmito datovými typy se projeví pouze v jednoduchostech, jako ve formátování syntaxe souřadnic do formátu pro daný datový typ. Co se však myšlenky a účelu těchto geometrií týče, tyto geometrie jsou identické.

Naproti tomu ostatní datové typy SQL Server Spatial nejsou zcela rozdílné, avšak ve většině případů umožňují reprezentaci několika specifických podmnožin datových prvků. Díky čemuž není konverze do jazyka GML zcela jednoznačná. „Následující objekty nelze zcela triviálně překonvertovat (většinou nemají jednoznačnou reprezentaci, případně je specifikace dosti vágní):“ [8]

### 4.3.2 Mapování geometrií z GML do SQL Server Spatial

Geometrie jazyka GML se na první pohled jeví jako daleko komplexnější. Je tomu tak hlavně proto, že jazyk GML disponuje množstvím zápisů geometrií, které tak z pohledu jazyka GML tvoří nové datové typy. Z hlediska cílů této práce je tento směr mapování velice důležitý, jelikož je potřebné nějak vyřešit ukládání geometrií obsažených ve vstupních souborech formátu AIXM 5.1. V případě identických datových typů je tato konverze oboustranně triviální. U ostatních geometrií je však nutné nejen zvolit adekvátní datový typ, ale také nějakým způsobem reflektovat možné odlišnosti ve struktuře reprezentace těchto typů. Pro přehlednost bylo mapování v rámci [8] rozděleno do jednotlivých kategorií podle knihoven.

**geometryBasic0d1d.xsd**

všechny tyto datové typy jsou obousměrně převeditelné.

**geometryPrimitives.xsd**

Datové typy *ArcByBulge*, *Arc*, *ArcByCenterPoint* a *CircleByCenterPoint*, včetně typů *ArcString*, *ArcStringByBulge* lze validně překonvertovat do SQL datového typu **CircularString**. Datový typ *LineStringSegment* lze pak překonvertovat do SQL datového typu **CompoundCurve**, případně do typu **LineString**.

**geometryComplexes.xsd**

Datové typy *CompositeCurve* lze konvertovat do SQL datového typu **CompoundCurve. Datové typy CompositeSurface** lze v případě použití reprezentace dat v podobě SQL třídy *Geography* konvertovat na **CurvePolygon či Polygon**. Třída *CompoundCurve* je pak svoji definicí nápadně podobná SQL datovému typu **GeometryCollection.**

# 5. Návrh datového modelu pro SQL server

Následující kapitola pojednává o možnostech návrhu datového modelu, který by byl schopen obstojně uchovávat vstupní data z formátu AIXM 5.1.

## 5.1 Klíčové vlastnosti standardu AIXM 5.1

Následující kapitola pojednává o vlastnostech standardu AIXM 5.1 a předkládá klíčové vlastnosti jazyka. Některé z těchto vlastností je možné použít pro optimalizaci datového modelu a zefektivnění případných operací nad ním.

Jak je uvedeno v kapitole AIXM 5.1., základní myšlenkou standardu AIXM 5.1 je reprezentace geografických objektů pomocí tzv. „*Features*“ a relacemi mezi nimi. V kapitole AIXM 5.1. byl také představen princip temporálního modelu a prvek *TimeSlice*. Rovněž zde byly také představeny prvky *Property*, včetně jejich realizací v podobě prvků *Attribute, Relationship* a *Object*. Všechny tyto prvky

jsou pro schéma klíčové a je třeba je nějak vhodně reprezentovat v databázi, při dodržení všech jejich podmínek.

### 5.1.1 Řešení referencí v AIXM 5.1

Reference v AIXM 5.1 je možné v databázovém modelu vyřešit zcela elegantně za pomoci cizích klíčů.

## 5.2 Ukládání XML dat do databáze

Dokumenty XML jsou vhodnou alternativou ke klasickým neuspořádaným plain–textovým souborům. Na rozdíl od těchto souborů je možné data v XML dokumentu strukturovat, a tím umožnit lepší čitelnost. Takto strukturované dokumenty jsou rovněž dobře čitelné pro programy a aplikace. Z tohoto důvodu také většina programů využívá značkovacího jazyka XML pro tvorbu a ukládání svých konfigurací do konfiguračních souborů, případně k ukládání svých dat do datových souborů. Nevýhodou XML dokumentů zůstává jejich neefektivní velikost. Množství elementů vytváří nadbytečný balast, který je nutné uchovávat. V případě přenosu souborů se může jednat o nadbytečná data, která je nutné přenést.

Co se týče přenosu dat, jsou soubory XML v tomto případě rozumnou volbou. Problém nastane v případě nutnosti operací nad těmito soubory. V případě nutnosti nalezení prvku s odpovídajícími vlastnostmi je v nejhorším případě, při triviálním přístupu k průchodu daty, nutné sekvenčně projít celý soubor, což značně prodlužuje dobu operace. Naproti tomu v případě použití databáze, coby úložiště pro data, může, při správné volbě typu databáze a reprezentace v databázi, dojít k rapidnímu zrychlení většiny operací. Zejména při operacích slučování (JOIN) či sjednocení (UNION) není třeba filtrovat jednotlivé vyhovující prvky, ale využít robustních konstrukcí a vestavěných algoritmů databázových programů. Důležitým aspektem je také podpora transakcí u relačních databází.

Z hlediska reprezentace XML dat v databázi je možné volit mezi několika přístupy. „V množství různých návrhů a projektů, které se tímto tématem zabývají, můžeme vypozorovat několik základních strategií ukládání XML dat: uložení s využitím systému souborů, uložení v relačním databázovém systému a nativní (přímé) uložení dat“ [3]

### 5.2.1 Databáze nad systémem souborů

V případě uložení s využitím systému souborů dochází k ukládání dat v klasickém lidsky čitelném formátu. Jedná se o klasické soubory ukládané v rámci souborového systému operačního systému. Logiku nad dokumenty pak obstarávají databázové programy. Vylepšením oproti triviálnímu přístupu je lepší podpora dotazů nad XML dokumenty. „Pro vyhodnocení dotazu tyto systémy obvykle nejprve vyhledají, načtou a analyzují všechny potřebné XML soubory. Veškerá XML data jsou následně uložena do speciálních paměťových struktur – stromů API DOM.“ [3] Tento přístup se jeví jako jednoduchý a elegantní, avšak po celou dobu dotazování je nutné, díky DOM reprezentaci, držet veškerý obsah souborů v paměti. Z tohoto hlediska mohou být paměťové nároky enormní.

### 5.2.2 Relační databáze

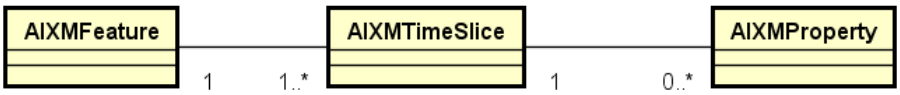
Dalším přístupem je ukládání dat do relační databáze. Tento přístup nemusí být pro některé případy ideální. V případě nutnosti výběru celého XML dokumentu si může generování struktury XML dat klást velké časové nároky. Naproti tomu v případě potřeby dotazování se na jednotlivé malé části XML dokumentu se jeví relační databáze jako ideální nástroj. Další výhodou je podpora transakčního zpracování, které zaručuje konzistenci dat během operací. Významným časovým zrychlením je také podpora indexace hodnot, či jednotlivých elementů napříč hierarchií dokumentu. V neposlední řadě také vhodná databázová struktura napomůže k dramatickému snížení velikosti uchovávaných dat.

## 5.3 Návrh datového modelu

Tato kapitola předkládá návrh možného datového modelu. Při designu tohoto modelu byl brán zřetel na optimalizaci datového modelu.

### 5.3.1 Základní entity

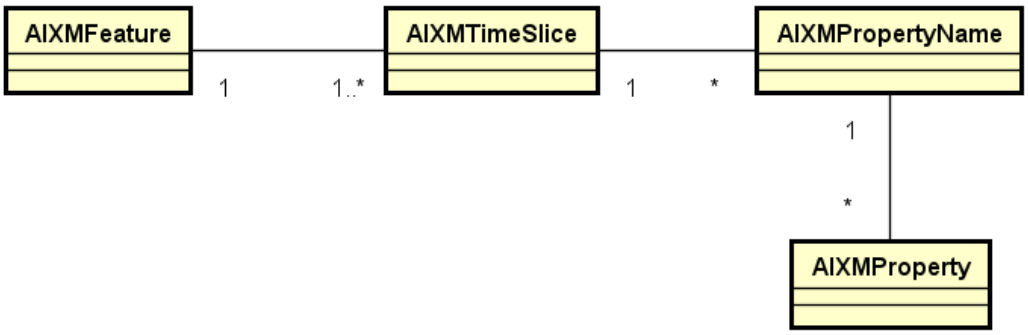
Jak bylo popsáno v kapitole o struktuře AIXM message – 3.7 AIXM 5.1 dokument. Klasická AIXM zpráva se skládá z kořenového elementu *AIXMBasicMessage*, který obsahuje jednotlivé elementy zprávy (element *hasMember*). Každý z těchto elementů obsahuje definici prvku *Feature*, která zahrnuje globální identifikátor tohoto prvku. Uvnitř tohoto elementu pak následuje element *TimeSlice*, který popisuje aktuální stav prvku *Feature* pomocí prvků, souhrnně nazývaných jako *Property*. Je patrné, že na první pohled je třeba brát v úvahu přítomnost prvků *Feature*, *TimeSlice* a *Property* jako samostatných entit v databázovém modelu. V průběhu času může mít prvek *Feature* několik prvků *Timeslice*, avšak prvek *TimeSlice* je výhradně podřízen právě jednomu prvku *Feature*. Prvek *TimeSlice* může mít několik prvků *Property*. Prvek *Property* je však podřízen právě jednomu prvku *TimeSlice*. Pro ilustraci následuje stručný class diagram.




Obrázek 9: Diagram základních entit

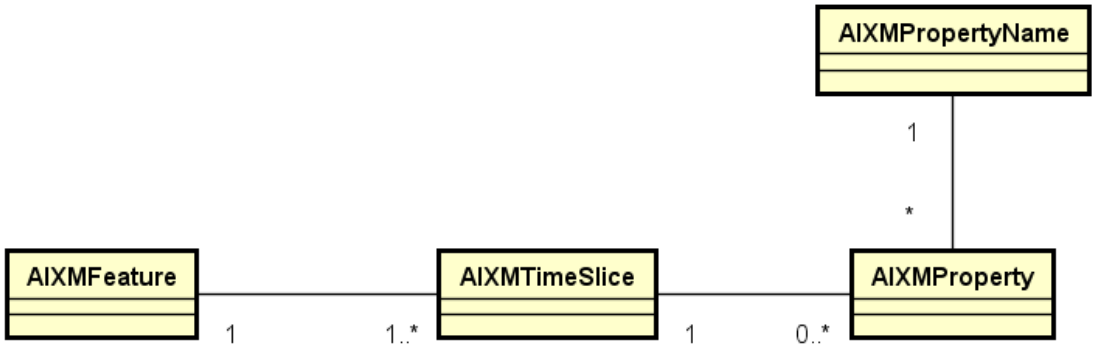
Při pohledu na data je na první pohled patrné, že jednotlivé prvky *Property* obsahují název a hodnotu. Co se týče názvu, je dosti pravděpodobné, že se jednotlivé názvy prvků budou v dokumentu opakovat. V případě dlouhých názvů je tak zbytečné uchovávat tyto názvy v databázi duplicitně. Z tohoto důvodu se zdá být lepším řešením rozdělit entitu *Property* na název prvku *Property* a jeho hodnotu. Takovéto rozdělení je možné uskutečnit několika způsoby:

1) Entita *TimeSlice* bude obsahovat seznam prvků (*PropertyName*) obsahující názvy prvků *Property*. Tyto entity budou obsahovat odkazy na všechny entity *Property* obsahující hodnoty.




Obrázek 10: Diagram řešení názvů Property přímo pod AIXMTimeSlice

2) Entita *Property* bude obsahovat pole s odkazem na entitu reprezentující název




Obrázek 11: Diagram řešení názvu Property v rámci pole v AIXMProperty

Vzhledem k množství druhů prvků *Property* je nutné počítat s různými formáty dat. Bylo by samozřejmě možné jednotlivá data ukládat do jednoho unifikovaného pole ve zdrojovém formátu, takovéto řešení by však bylo značně neoptimalizované a také by značně zesložiťovalo tvorbu případných dotazů. Z tohoto důvodu byla navržena strategie dědičnosti pomocí tzv. *Single Table*. Tato strategie spočívá v ukládání hodnot jednotlivých druhů *Property* do stejné tabulky. Jednotlivé druhy dat jsou pak rozeznány na základě jednoznačného identifikátoru (v databázi SQL Server nazýván jako tzv. „*Discriminator column*“).

Další rozšíření spočívá v možné hloubce jednotlivých prvků. V rámci dat AIXM 5.1 se nezřídka stává, že pro lepší kompaktnost dochází k jakémusi „obalování“ hodnot prvků *Property* do dalších struktur. Povětšinou se jedná pouze o další styl formátování XML dokumentu, existují však případy, jako například struktura datumu, kdy je vhodné si tuto strukturu zapamatovat. Z tohoto důvodu byl do databázového modelu přidán, do entity *Property*, cizí klíč, který odkazuje na předka konkrétní instance *Property*. Předkem entity *Property* může být výhradně jen instance entity *Property*. V případě vrcholové instance, přímo podřízené entitě *TimeSlice* je toto pole prázdné (nabývá hodnoty NULL).

V případě výskytu prvku *Attribute* v dokumentu se tato hodnota tohoto prvku uloží do pole value. V dokumentu se však může vyskytnout také varianta, která obsahuje číselnou hodnotu a také přidaný

atribut s informací o jednotce dané hodnoty. Tato entita si zaslouží být oddělena od prvku *Attribute*. Dostala název *AttributeWithUom*.

V případě výskytu objektu je nutno tento objekt celý uložit. K tomuto bylo využito přítomnosti rekursivního cizího klíče v prvku *Property*. Výsledná reprezentace má podobu objektu *Object* (vrcholná entita *Propertry*) a *SubObject* reprezentující jednotlivé vlastnosti objektu.

Jako další z možných instancí prvku *Property* se může v dokumentu vyskytnout instance geometrie. V takovémto případě je třeba nového datového typu pro prvek *Property*. Jednou z možností je umístit do entity *Property* další sloupec s datovým typem geometrie. V praxi by však mohlo být výhodné pracovat pouze s tabulkou geometrií. Reprezentace geometrie byla proto uvedena do statutu samostatné entity z důvodu lepší dostupnosti. Samotná entita *Property* pak, v případě přítomnosti geometrického prvku, obsahuje cizí klíč na entitu *GeometryProperty*.

Jak již bylo psáno dříve, referenci na prvek *Feature* je možné pojmout formou cizího klíče. Pro logické uspořádání však byla vytvořena doplňující entita *Relationship*, které jednoznačně určuje typ *Property*.

### 5.3.2 Rozšíření entity Property

Na základě zkoumání dodaných vstupních souborů a oficiálních XSD schémat byly vytipovány následující datové typy:

- **Attribute** – klasický druh elementu mající prostou textovou hodnotu
- **AttributeWithUom** – rozšířený druh atributu, který obsahuje číselnou hodnotu doplněnou o definici jednotky
- **Relationship** – druh *Property*, který realizuje vazby mezi AIXM features (prvky Relationships); obsahuje referenci na entitu *Feature*
- **AbstractObject** – tento prvek reprezentuje AIXM prvek *Object*
- **Object** – prvek reprezentující jeden *AIXMObject*
- **ObjectCollection** – prvek reprezentující kolekci prvků *AIXMObject*
- **ObjectProperty** – reprezentuje jednotlivé vlastnosti objektu
- **ValidTime** – jedná se o jeden z důležitých prvků prvku *TimeSlice*, volba na samostatnou entitu byla provedena z důvodu celkové komplexnosti vzorové struktury
- **FeatureLifeTime** – další z důležitých prvků prvku *TimeSlice*, volba samostatnosti byla provedena ze stejného důvodu jako v případě entity *ValidTime*
- **GeometryObject** – tato entita reprezentuje element s geometrií. Přítomnost geometrie *GeometryProperty* je záměrně uvedena ve formě cizího klíče do tabulky s geometriemi. Tento přístup byl zvolen z důvodu jasného oddělení geometrických útvarů od zbytkových převážně textových dat

### 5.3.3 Entita GeometryProperty

Pro entitu Geometrie byl zvolen datový typ geography. Jak již bylo pojednáno v kapitolách 4.1 Geometrie v Microsoft SQL server spatial, tento datový typ umožňuje ukládání geometrie do sférických souřadnic, což je v přímé shodě s povahou dat v AIMX 5.1.

## 5.4 Souřadnicové systémy v SQL Server Spatial

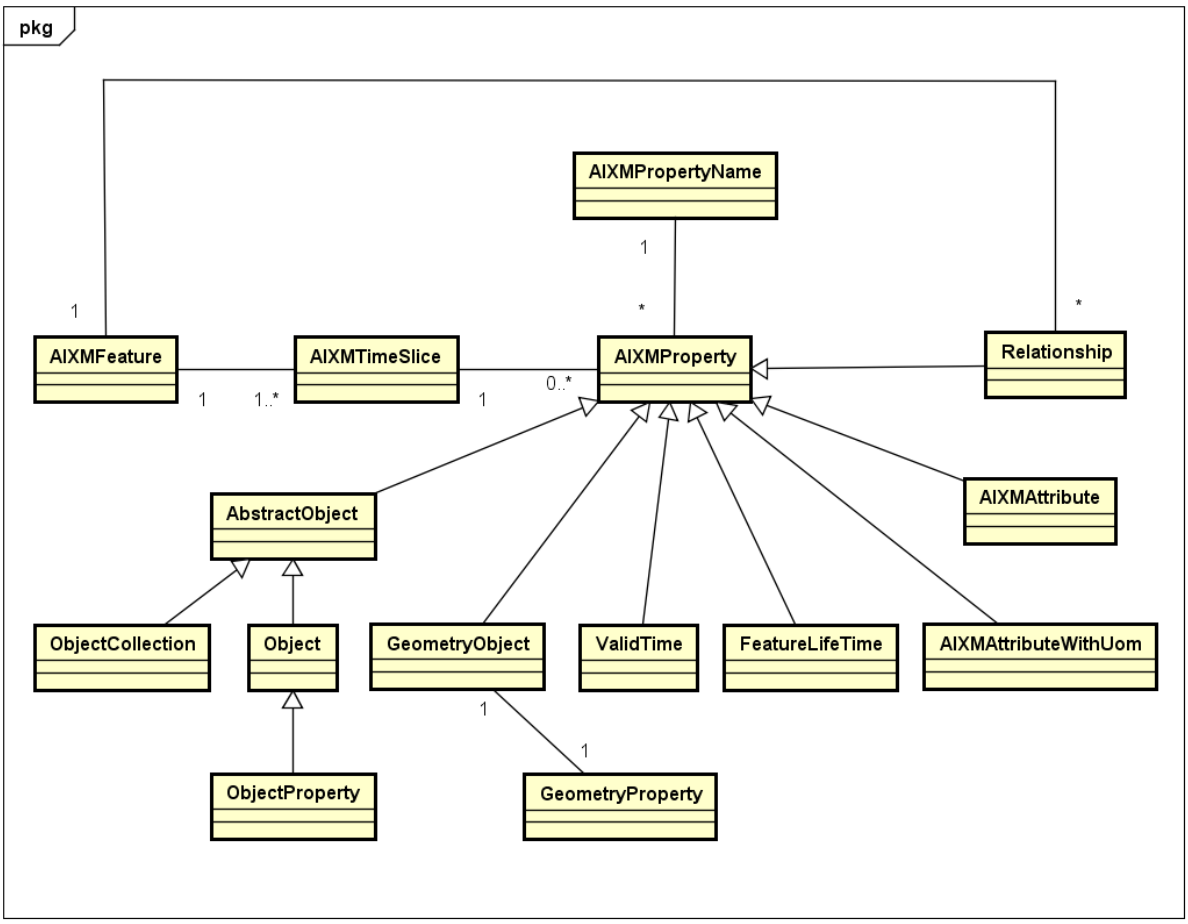
Souřadnicové systémy jsou v Microsoft SQL Server Spatial reprezentovány pomocí tzv. *SRID* identifikátoru. O významu tohoto identifikátoru bylo pojednáno v úvodu práce v kapitole 1.1.4 SRID identifikátor. V rámci výzkumného projektu [8] byla zjištěna klíčová informace o SQL Server Spatial,

se kterou je třeba v implementaci počítat: „Dvě Geography instance s různými *SRID* nemohou být porovnány.“ Jednotlivá data je tak třeba buďto konvertovat do jednotného formátu, nebo s vědomím různých souřadnicových systémů dotazy provádět nad konkrétními podmnožinami dat. Veškeré možné reprezentace dat je v případě Microsoft SQL Serveru Spatial možné zjistit následujícím SQL příkazem:

```
SELECT * FROM sys.spatial_reference_systems
```

**Výpis 6: Příkaz ke zjištění dostupných projekčních systémů**

Závěrem této kapitoly je pro přehlednost uveden kompletní UML diagram databázové struktury, který zobrazuje jednotlivé entity ve formě tříd, tak jak by se měly objevit v objektově relačním mapování aplikace.




**Obrázek 12: Diagram všech databázových entit**

# 6. Modul pro import a export AIXM 5.1 dat

Na základě zmapování stadnardu AIXM 5.1, jazyka GML a geometrií Microsoft SQL Server Spatial byl vytvořen program, který je schopen naparsovat vstupní AIXM 5.1 zprávu a uložit její údaje do databáze. Následující kapitola popisuje postup při tvorbě tohoto programu. Celou problematiku tvorby by bylo možné rozdělit do jednotlivých kategorií:¨

- Parsování vstupního souboru
- Reprezentace v programu
- Ukládání do databáze

## 6.1 Přehled existujících technologií

Následuje přehled možných technologií, které by mohly být užitečné pro implementaci jednotlivých částí programu.

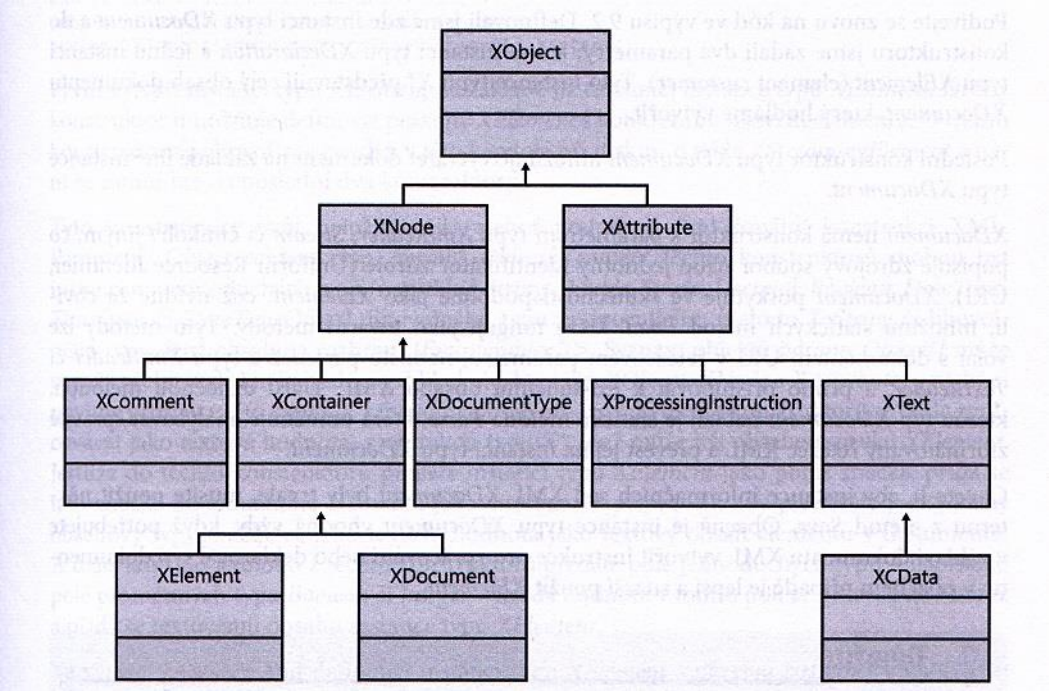
### 6.1.1 Parsování XML dat

K parsování XML dokumentů existuje množství nástrojů a knihoven. Tyto technologie by se daly rozdělit na dvě kategorie a to na Sekvenční parsery a tzv. DOM parsery. V případě Sekvenčních parserů je vstupní soubor čten sekvenčně. Tento způsob si neklade vysoké nároky na paměť, avšak vzhledem k sekvenčnímu čtení neumožňuje návrat na již přečtené části dokumentu. Problémem by tak mohlo být řešení referencí v dokumentu. Naproti tomu tzv. DOM parsery načtou celý obsah souboru do paměti a je pak možné nad ním provádět dotazy. Tento přístup je jeví jako dobrý, avšak pro velké soubory takřka nepoužitelný z důvodu již zmíněné nutnosti nahrát celý soubor do paměti.

Následují příklady parserů:

#### *6.1.1.1 Linq to XML*

Jedná se o knihovny frameworku LINQ specializující se na práci s daty XML. Jednotlivé elementy XML dokumentu jsou reprezentovány pomocí tříd. Tato knihovna umožňuje načtení obsahu celého, či části dokumentu do hierarchické, stromové reprezentace pomocí instancí tříd. Následně díky podpoře syntaxe z LINQ umožňuje tvorbu dotazů nad touto strukturou. Umožňuje kompletní podporu CRUD operací nad daty. Z výše zmiňovaných typů parserů se řadí ke katerogii DOM parserů.




Obrázek 13: Přehled struktury objektu Linq To XML

#### *6.1.1.2 XMLReader*

Třída XmlReader je klasická robustní třída, pro práci XML dokumenty. Umožňuje sekvenčně číst vstupní XML dokument. Z výše jmenovaných typů parserů se řadí ke kategorii Sekvenčních parserů.

### 6.1.2 ORM

Pro objektově relační mapování byly vytipovány následující frameworky:

- **Linq to SQL** – framework, vyznačující se přímou podporou připojení k SQL serveru
- **Entity Framework** – původní součást .NET frameworku
- **NHibernate** – open source framework (mutace frameworku Hibernate pro prostředí .NET)

Hlavním kritériem pro výběr finálního ORM frameworku byla kvalitní podpora geografických typů Microsoft SQL Serveru.

#### *6.1.2.1 Linq to SQL*

Tento framework se vyznačuje přímou podporou základních operací nad databází SQL Server. Jeho význam spočívá v přesném mapování 1:1 vůči databázovým tabulkám.

Hlavní nevýhodou použití pro účely této práce je celková zastaralost frameworku [34] a absence knihoven pro práci s prostorovými daty (Framework neumožňuje mapování geografických typů DbGeography a SQLGeography).

#### *6.1.2.2 Entity Framework*

Entity framework je objektově relační mapovač, který umožňuje vývojářům na platformě .NET pracovat s relačními daty s použitím doménově specifických objektů. [35] První verze byly vyvíjeny a také dodávány jako součást prostředí .NET Framework, posléze však Microsoft uvolnil zdrojové kódy a od verze 6 se tak Entity Framework stává open source projektem. [36] [37] Po implementační stránce umožňuje entity Framework zvolit z několika možných scénářů tvorby entit. Jedním kritériem je volba mezi programátorským přístupem tzv. CodeFirst a přístupem designerským tzv. „“. V případě CodeFirst přístupu se pracuje s anotacemi a mapováním relací v rámci kódů. V případě „“ se databáze tvoří pomocí grafikých diagramů. I v tomto případě lze v projektu nalézt skryté soubory se zdrojovými kódy pro mapování entit, avšak tento přístup je výhradně orientován na modelování přes diagramy – veškeré kód tak generuje prostředí.

Dalším kritériem je existence databáze. Je možné si vybrat z přístupu vytvoření nové databáze z kódu či diagramu, nebo generování entit na základě již existující databáze. Bližší informace je možné dohledat na oficiálních stránkách produktu. [38]


Obrázek 14: Přehled možných přístupů k databázi (dostupné z: [38])

#### 6.1.2.3 NHibernate

Projekt Nhibernate vychází z originálního projektu Hibernate, který je hojně využíván v prostředí jazyka Java. NHibernate pak představuje jakousi mutací do prostředí jazyka C# a platformy .NET obecně.

### 6.1.3 Parsování Geometrií do Microsoft SQL Server Spatial

Microsoft SQL server spatial sám o sobě disponuje užitečnými funkcemi pro práci s GML. Pro účely „parsování" dat disponuje SQL Server Spatial metodou *geometry::GeomFromGml()*. Tato metoda vezme validní vstup ve formátu GML a přetransformuje jej na instanci typu geography. (viz.Výpis 7)

```
DECLARE @g geography;
DECLARE @x xml;
SET @x = '<LineString
xmlns="http://www.opengis.net/gml"><posList>47.656 -122.36 47.656 -
122.343</posList></LineString>';
SET @g = geography::GeomFromGml(@x, 4326);
SELECT @g.ToString();
```

```
DECLARE @g geography;
DECLARE @x xml;
SET @x = '<FullGlobe
xmlns="http://schemas.microsoft.com/sqlserver/2011/geography" />';
SET @g = geography::GeomFromGml(@x, 4326);
SELECT @g.ToString();
```

**Výpis 7: Ukázka příkazu pro parsování GML geometrií**

Pro dokonalou konverzi do GML SQL server spatial disponuje metodou *.AsGML()*. Tato metoda, aplikovaná na geografická data, převede tyto data do reprezentace v jazyku GML. Ze zběhlého pozorování a testování lze usoudit, že tato metoda jest napsána správně a všechny instance korektně převádí do validního GML. (viz. Výpis 8)

```
DECLARE @g geography;
SET @g = geography::STGeomFromText('LINESTRING(-122.360 47.656, -
122.343 47.656)', 4326);
SELECT @g.AsGml();
```

```
<LineString xmlns="http://www.opengis.net/gml"><posList>47.656 -
122.36 47.656 -122.343</posList></LineString>
```

**Výpis 8: Ukázka příkazu pro zpětný výpis do GML**

Tyto metody se samy o sobě jeví jako dosti užitečné. Bohužel, při bližším zkoumání, byl zjištěn fakt, že tyto metody podporují pouze určitou podmnožinu jazyka GML. Není proto možné obsáhnout veškeré geometrie obsažené v GML 3.2. Seznam těchto geometrií je uveden v [39].

Jako další nevýhoda těchto metod byla zjištěna podpora pro převody, pouze dvojrozměrných dat. V případě metody *geometry::GeomFromGml()* dojde k vyhození chybové hlášky a převod se neprovede. U metody *AsGml()* ke konverzi do GML dojde, avšak samotná metoda bez jakéhokoli varování zahazuje všechny souřadnice vyšších rozměrů než dvojrozměrného. Z těchto důvodů je

možné tyto metody použít, avšak pouze na jednoduchá dvojrozměrná data, splňující podporované GML datové typy.

Další možností vkládání geometrií do Microsoft SQL server spatial je použití textového formátu (tzv. Well-Know text). Tento formát byl již popsán v kapitole 1.4.2 Well-known Text.

## 6.2 Návrh modulu

Tato kapitola pojednává o návrhu struktury výsledné aplikace. Veškeré návrhy v této kapitole byly následně zohledněny při tvorbě finální aplikace.

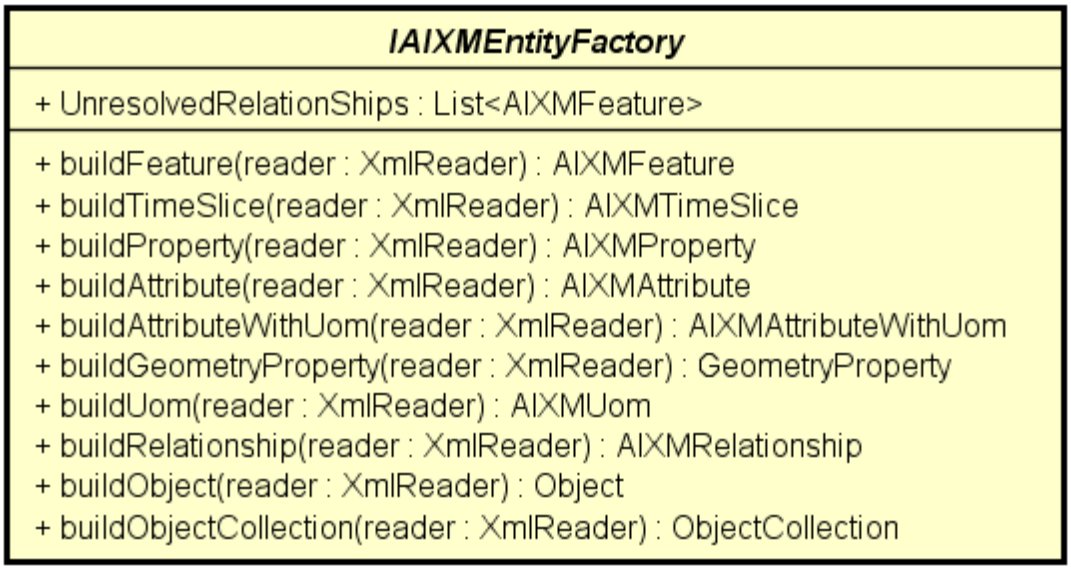
### 6.2.1 Parsování dat

Vstupní data programu budou obsažena v klasickém XML dokumentu. Je tedy třeba navrhnout třídu, která vezme adresu vstupního souboru a vrátí seznam *AIXMFeature* elmentů v něm obsažených. Tuto třídu si lze představit pod následujícím obrázkem:


Obrázek 15: Diagram třídy pro parsování dat

K parsování dat byl vymyšlen přístup, který staví na návrhovém vzoru *Factory*. [42] V případě nalezení elementu který odpovídá AIXM prvku *Feature* dojde k zavolání metody *buildFeature()*, parametrem metody je aktuální instance třídy *XmlReader*. Tato metoda by měla načíst veškeré informace potřebné k sestavení třídy *AIXMFeature*. V případě, že tato metoda narazí na element pro prvek *TimeSlice*, dojde k zavolání metody *buildTimeSlice()*. Tato metoda má obdobné chování jako metoda *buildFeature()*. V případě nalezení elementu odpovídajícímu AIXM prvku *Property* dojde k zavolání metody *buildProperty()* se stejným chováním jako předchozí „build" metody. Díky celkové mělkosti jednotlivých úrovní v AIXM dokumentu není třeba obávat se selháním v podobě přetečení zásobníku množstvím rekurzivních volání. Následující diagram ilustruje strukturu rozhraní pro sestavení jednotlivých AIXM prvků. Následující obrázek pak ilustruje na ukázkovém dokumentu, které části uvedeného AIXM dokumentu budou zpracovány kterou metodou. Za pozornost stojí elementy *interception*, *sequenceNumber* a *correctionNumber*, které jsou neoznačeny záměrně. Jedná se totiž o klíčové vlastnosti prvku *TimeSlice* – proto jsou tyto elementy parsovány již v metodě *buildTimeSlice()* a není tak pro ně třeba volání dalších metod.


Obrázek 16: Ukázka konzumace souboru jednotlivými metodami parseru



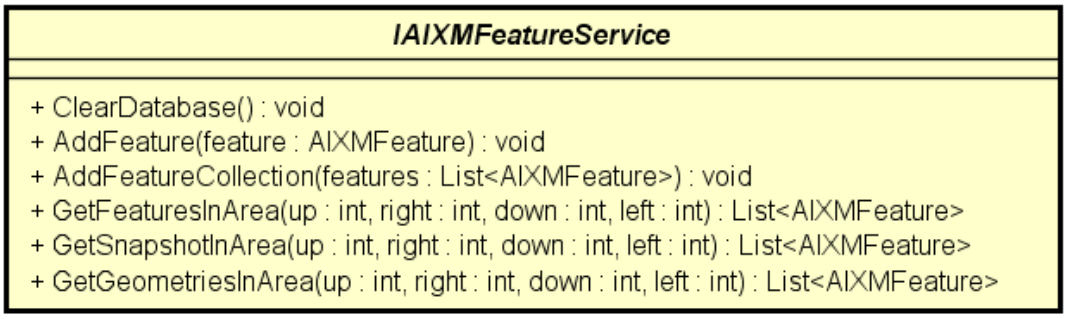

Obrázek 17: Diagram třídy pro tvorbu prvků Feature z AIXM 5.1

### 6.2.2 ORM

Pro ukládání dat byla zvolena persistentní vrstva spravovaná pomocí Entity Frameworku. Tato volba byla vybrána z důvodu donedávné součásti frameworku v platformě .NET.

### 6.2.3 Dotazování

K účelům dotazování bylo navrženo následující rozhraní:




**Obrázek 18: Diagram třídy pro přístup do databáze**

Umožňuje vkládání *AIXMFeature* prvků jak jednotlivě, tak v kolekci. Důležitou části jsou metody pro výběr prvků Feature na základě vytyčené plochy. Jako vstupní plocha byl zvolen klasický obdélník. Tento obdélník je definován čtveřicí bodů, reprezentující pozice jeho stran. Metoda *GetSnapshotInArea* vrací kolekci prvků *Feature*, která obsahuje aktuální hodnoty všech prvků *Property* (viz kapitola 3.3 Temporální model). Každý prvek *Feature* tedy obsahuje pouze jeden prvek *TimeSlice*. Metoda *GetFeaturesInArea* pak vrací kolekci prvků *Feature*, která však obsahuje veškeré hodnoty prvků *Property* v průběhu času (kategorizované dle prvků *Timeslice*).

V rámci výzkumného projektu [8] byly vytipovány použitelné dotazovací metody. Dotazování se na geografická data probíhá v SQL serveru obdobně jako tomu je u klasických relačních databází – za pomoci syntaxe jazyka SQL. Na rozdíl od klasického SQL však, pro výběry dat, disponuje SQL Server Spatial metodami pro práci s geometrickým prostorem.

Pro účely projektu byly vybrány následující metody:

**STArea()** – Tato metoda vrací obsah (objem) zvoleného polygonu.

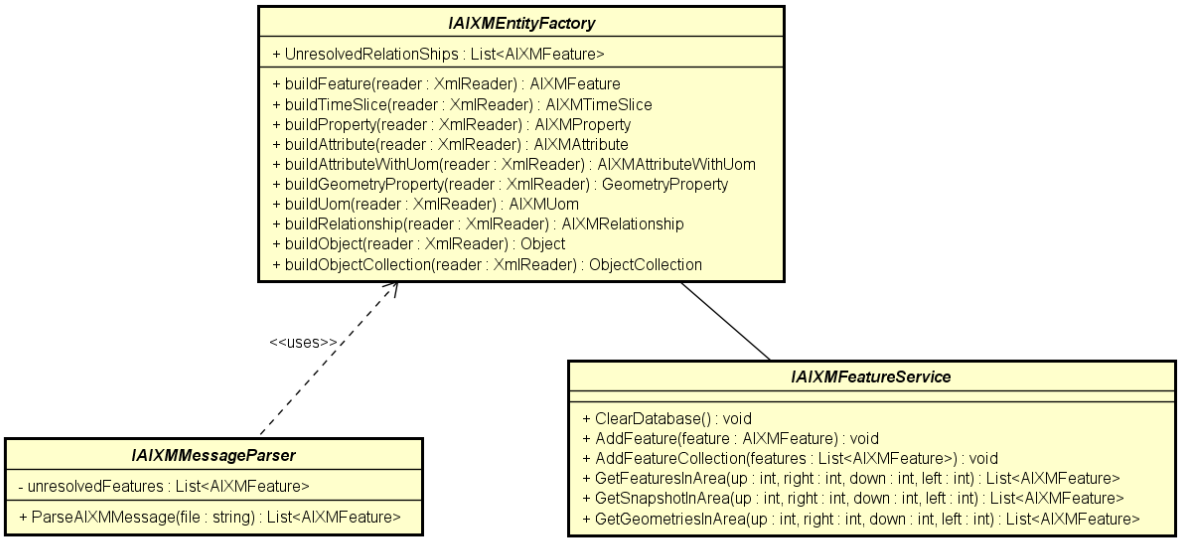
**STIntersects()** – Tato metoda provede průnik dvou geografických útvarů.

**STDistance()** – Metoda vrací nejkratší vzdálenost mezi zadanými body.

**STAsText()** – Zobrazí standardní textovou reprezentaci dat. Tento příkaz může najít uplatnění při testování a ladění.

**STIsValid()** – tato metoda zkontroluje, zdali zadaný útvar je validní. Tento příkaz se rovněž může hodit pro kontrolu správnosti zadávaných dat.

Závěrem je uveden souhrnný class diagram, který zobrazuje celkový pohled na modul.




Obrázek 19: Diagram celkového pohledu na návrh modulu

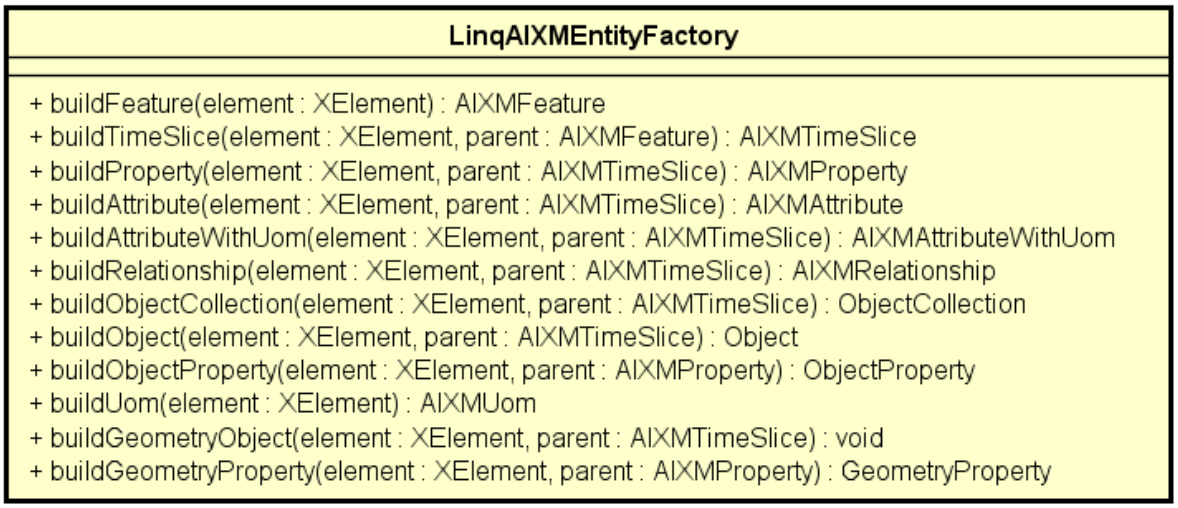
## 6.3 Implementace

Tato kapitola popisuje postupy a řešení použitá při implementaci modulu. K implementaci byl po konzultaci s vedoucím zvolen jazyk C#.

### 6.3.1 Parsování Dat

K parsování dat byla nakonec zvolena kombinace Sekvenčního čtení vstupního souboru s DOM parserem. Soubor je sekvenčně čten pomocí třídy *XmlReader*. V momentu stanutí na úrovní prvku *Feature* však dojde k načtení celého elementu do stromové reprezentace pomocí Linq to XML parseru. Díky této metodě je možné vybírat prvky napříč celou stromovou reprezentací prvku. V rámci dat dodaných vedoucím neexistovat prvek, který by svou velikostí přesahoval možnosti načtení Linq to XML parserem.

Tato stromová reprezentace je následně podrobena již známé rekurzivní proceduře z kapitoly 6.2 Návrh modulu, která analyzuje subelementy a vytváří příslušné třídy. Jedinou úpravou oproti rozhraní bylo vytvoření nových metod, které jsou schopny pracovat nad prvky Linq to XML parseru – *XElement*. Pro plnou kompatibilitu byly implementovány i ostatní metody rozhraní, prostým přemostěním, na jejich ekvivalenty pro parsování tříd *XElement*.




Obrázek 20: Diagram implementace třídy pro načítání dat

### 6.3.2 Datový model

Jako provider persistentní vrstvy byl zvolen Entity Framework verze 7. Tato volba byla provedena díky podpoře mapování geometrií v podobě datového typu *DBGeography*. Tento typ je sice obecnějším geometrickým typem (podporuje připojení do více databází), na druhou stranu právě tato informace zjednodušuje případné použití nad dalšími databázemi.

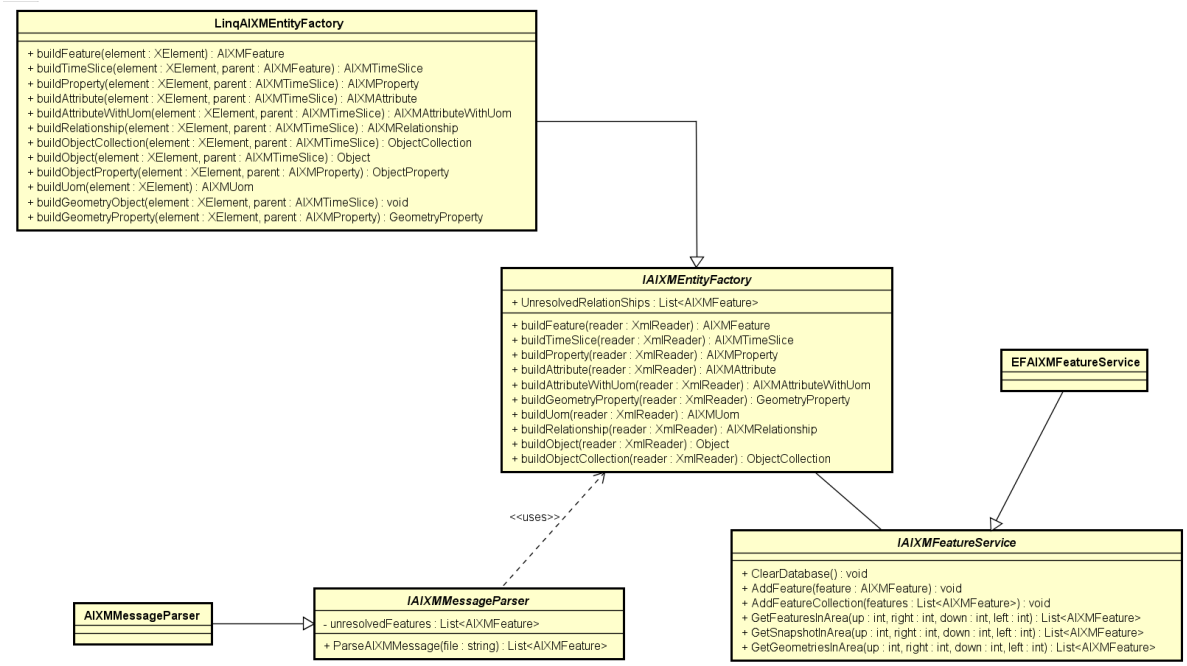
### 6.3.3 Reprezentace geometrické informace

Geometrická informace byla původně parsována za pomoci metody *FromGML()*, což je metoda, která následně volá známou metodu *STGeomFromText()* SQL Serveru. Po zjištění nedostatků této metody bylo celé parsování metod přetvořeno na parsování z formátu WKT (*Well-known text*). Důvodem k volbě tohoto přístupu byla možnost vytvoření instance třídy *DbGeography* z tohoto formátu pomocí metody *FromText()*, která stejně jako předchozí zmíněná metoda také volá svůj ekvivalent v podobě metody SQL Serveru *STGeomFromText()*. Na rozdíl od první použité obstojně podporuje vícerozměrná data. Z důvodu převodu GML geometrií do WKT byla vytvořena knihovna, pro generování WKT řetězců ze vstupního GML elementu. Vzhledem k povaze dodaných dat byla zvolena podmnožina jednotlivých geometrií. Struktura programu je však natolik robustní, že případné doplnění geometrií pohodlně umožňuje.

### 6.3.4 Detekce projekce

Pro detekci projekce byla rovněž vytvořena knihovna, která pro vstupní GML element zjistí použitou projekci. Jako další rys umožňuje identifikovat na základě definice projekce, v podobě *srsName* atributu, odpovídající *SRID projekci*. Byla rovněž napsána knihovna, která umožňuje převod z definovaných projekčních systémů do projekčního systému s SRID 4326.

Závěrem kapitoly je předložen diagram, který zobrazuje celkovou strukturu modulu. (Celý tento diagram v originální velikosti je součástí přílohy této práce – složka */diagramy*)




Obrázek 21: Diagram celkového pohledu na modul

# 7. Ověření funkčnosti modulu

V této kapitole je rozebrán postup, kterým byla ověřena funkčnost modulu. Závěrem jsou uvedeny výsledky měření a jejich interpretace.

## 7.1 Testovací data

Pro účely studia technologie byla vedoucím dodána datová sada obsahující data od společnosti Eurocontrol. Jednotlivé soubory jsou přítomny v rámci přílohy této práce, která je uložena na přiloženém disku v adresáři `/testdata`.

V rámci těchto souborů byla aplikace nejprve otestována na souboru s názvem `featureExample.xml`. Tento soubor obsahuje pouze jeden prvek typu *Feature*, který obsahuje prvek *TimeSlice* a několik prvků *Property*. Prvky *Property* v tomto dokumentu nejsou typu *Relationship*. Program tyto data načetl a úspěšně uložil do databáze. Následně bylo přikročeno k testu nad zbylými daty. V rámci těchto testů byl však zjištěn fakt, že některé soubory obsahují inkonzistentní data. Jednalo se zejména o nevyplněné povinné položky datumů u prvků *Property validTime* a *featureLifeTime*. Z těchto důvodů byly tyto soubory vyřazeny z měření.

## 7.2 Způsob měření

Na základě velikostí souborů byly vytipovány dva exempláře, které se svoji velikostí přibližují k obřím datovým zprávám AIXM. Klíčovým aspektem bylo ověření správného chodu programu. Tedy zdali je program schopen obstojně naparsovat vstupní soubor. Jako bonusový cíl bylo vytyčeno zmapování doby běhu programu s různými konfiguračními nastaveními. Tato konfigurace spočívala v použití různých režimů ukládání. Pro porovnání byla měřena zvlášť fáze parsování a ukládání.

## 7.3 Výsledky měření

Během měření vyplynulo, že právě ukládání do databáze trvá enormně dlouhou dobu. Tomuto faktu je přikládáno za vinu hlavně použití Entity Frameworku, který se ukázal být značně těžkotonážním a

pomalým. K určitému, ne však markantnímu zlepšení ve výkonu přispělo vypnutí vlastnosti `Configuration.ValidateOnSaveEnabled`, která validuje případné změny v prvcích. I tak bylo ukládání do databáze nejslabším prvkem modulu.

# 8. Rozvoj do budoucna

V rámci dalšího rozvoje by bylo výhodné zaměřit se na databázovou část, přesněji na výkonnost persistentního modulu. Zajímavým porovnáním by bylo porovnání ukládání persistentního frameworku s klasickým vkládáním dat do databáze pomocí jazyka SQL.

# 9. Závěr

V rámci práce byly zmapovány jednotlivé geometrické jazyky včetně dvou klíčových AIXM 5.1 a GML. Následně byl porovnán systém geometrií Microsoft SQL Server Spatial a jazyka GML. Došlo ke zjištění, že většina geometrií je oboustranně převeditelná ať už triviálně, či za pomoci nějaké konverze. Následně byla navržena struktura modulu pro import AIXM 5.1 dat. Tato struktura pak posloužila k implementaci modulu v jazyce C# za použití Entity Frameworku coby persistentního providera a Linq To Xml a třídy *XmlReader* coby kombinované techniky pro parsování vstupních dat. Výsledný modul byl podroben ověření schopnosti pracovat nad aeronautickými daty, kde uspěl, avšak byla zjištěna pomalá odezva EntityFrameworku. Závěrem je třeba podotknout, že aplikace je teprve ve stádiu raného akademického vývoje a až podrobné testování ukáže, zdali je schopna běhu v reálném prostředí.

# Seznam literatury


[1] RON LAKE .. [ET AL.]. *Geography mark-up language (GML)*. Repr. Chichester: John Wiley, 2004. ISBN 978-047-0871-546.

[2] PIALORSI, Paolo a Marco RUSSO. *Microsoft LINQ: kompletní průvodce programátora*. Brno: Computer Press, 2009. Programování (Computer Press). ISBN 978-80-251-2735-3.

[3] HOLUBOVÁ, Irena, Karel RICHTA, Martin NEČASKÝ, Kamil TOMAN a Vojtěch TOMAN. *XML technologie: principy a aplikace v praxi*. Praha: Grada, 2008, 267 s. Průvodce (Grada). ISBN 978-80-247-2725-7.

[4] SCHNEIDER, Markus. *Spatial data types for database systems: finite resolution geometry for geographic information systems*. New York: Springer, 1997. ISBN 35-406-3454-1.

[5] HART, Glen a Catherine DOLBEAR. *Linked data: A Geographic Perspective*. Boca Raton: Taylor, 2013. ISBN 978-143-9869-956.

[6] Geography Markup Language. *OGC: Open Geo Spatial* [online]. 2016 [cit. 2016-05-25]. Dostupné z: http://www.opengeospatial.org/standards/gml. GML Specification.

[7] *OGC: Open Geospatial Consorcium* [online]. 2016 [cit. 2016-05-25]. Dostupné z: http://www.opengeospatial.org

[8] ŠROGL, Jan. *Zpráva o projektu: [A4M33SVP] Softwarový nebo výzkumný projekt*. České vysoké učení technické Fakulta elektrotechnická. 2016.

[9] World Geodetic System. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-05-24]. Dostupné z: https://en.wikipedia.org/wiki/World_Geodetic_System

[10] HOFMAN, Martin. *Ukládání geografických dat v NoSQL databázi*. Praha, 2015. Dostupné také z: https://dspace.cvut.cz/bitstream/handle/10467/62065/F3-BP-2015-Hofman-Martin-Ukladani_geografickych_dat_v_NoSQL_databazi.pdf?sequence=2&isAllowed=y. Bakalářská práce. České vysoké učení technické v Praze Fakulta elektrotechnická.

[11] Coordinate Reference Systems. W3C: World Wide Web Consortium [online]. 2015 [cit. 2016-05-26]. Dostupné z: https://www.w3.org/2015/spatial/wiki/Coordinate_Reference_Systems

[12] European Terrestrial Reference System 1989. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-05-25]. Dostupné z: https://en.wikipedia.org/wiki/European_Terrestrial_Reference_System_1989

[13] *EPSG Geodetic Parameter Registry: Version: 8.9.2* [online]. [cit. 2016-05-26]. Dostupné z: http://www.epsg-registry.org

[14] *EPSG* [online]. International Association of Oil & Gas Producers, 2016 [cit. 2016-05-26]. Dostupné z: http://www.epsg.org/

[15] ZHONG, Zhi-Nong, Ning JING, Luo CHEN a Qiu-Yun WU. Representing topological relationships among heterogeneous Geometry-Collection features. Journal of Computer Science and Technology [online]. 2004, 19(3), 280-289 [cit. 2016-03-15]. DOI: 10.1007/BF02944898. ISSN 1000-9000. Dostupné z: http://link.springer.com/10.1007/BF02944898

[16] DE-9IM. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-05-26]. Dostupné z: https://en.wikipedia.org/wiki/DE-9IM
[17] CLEMENTINI, Eliseo, Paolino FELICE a Peter OOSTEROM. A small set of formal topological relationships suitable for end-user interaction [online]. s. 277 [cit. 2016-03-15]. DOI: 10.1007/3-540-56869-7_16. Dostupné z: http://link.springer.com/10.1007/3-540-56869-7_16
[18] *GEOJSON* [online]. 2016 [cit. 2016-05-26]. Dostupné z: geoJson.org
[19] LOTT, Roger (ed.). Geographic information - Well-known text representation of coordinate reference systems.*Open Geospatial Consortium* [online]. 2015 [cit. 2016-05-26]. Dostupné z: http://docs.opengeospatial.org/is/12-063r5/12-063r5.html
[20] ST_AsText. *PostGIS 1.4 Manual* [online]. [cit. 2016-05-26]. Dostupné z: http://postgis.net/docs/manual-1.4/ST_AsText.html
[21] STGeomFromText: (geometry Data Type). *Microsoft Developer Network* [online]. Microsoft Corporation, 2016 [cit. 2016-05-26]. Dostupné z: https://msdn.microsoft.com/en-us/library/bb933823.aspx
[22] Well-known text (WKT) representation: Supported data formats. *IBM Knowledge Center* [online]. International Business Machines Corporation [cit. 2016-05-26]. Dostupné z: http://www.ibm.com/support/knowledgecenter/SSEPGG_9.7.0/com.ibm.db2.luw.spatial.topics.doc/doc/rsbp4120.html
[23] OGC Standards: Below is a list of OGC Implementation Standards. *OGC: Open Geospatial Consorsium* [online]. Open Geospatial Consorsium, 2016 [cit. 2016-05-26]. Dostupné z: http://www.opengeospatial.org/docs/is/?page=specs
[24] Simple Features. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-05-26]. Dostupné z: https://en.wikipedia.org/wiki/Simple_Features
[25] AIXM 5.1 Specification. *Aeronautical Information Exchange Model* [online]. EUROCONTROL, 2016 [cit. 2016-05-26]. Dostupné z: http://www.aixm.aero/page/aixm-51-specification
[26] PORTELE, Clemens (ed.). *OpenGIS® Geography Markup Language (GML) Encoding Standard: Version: 3.2.1*[online]. In: OPEN GEOSPATIAL CONSORTIUM INC. 2007, s. 437 [cit. 2016-05-26]. Dostupné z: http://portal.opengeospatial.org/files/?artifact_id=20509
[27] Aeronautical Information Exchange. EUROCONTROL. AIXM: Aeronautical Information Exchange [online]. 2016 [cit. 2016-01-22]. Dostupné z: http://www.aixm.aero/public/subsite_homepage/homepage.html
[28] Introduction. AIXM: Aeronautical Information Exchange [online]. 2016, 2006-12-15 [cit. 2016-01-22]. Dostupné z: http://www.aixm.aero/public/standard_page/introduction.html
[29] Class - AirportHeliport. *AIXM Wiki - Home* [online]. EUROCONTROL, 2010, 2010/02/19 14:01 [cit. 2016-05-26]. Dostupné z: https://ext.eurocontrol.int/aixmwiki_public/bin/view/AIXM/Class_AirportHeliport
[30] Class - Airspace. *AIXM Wiki - Home* [online]. EUROCONTROL, 2010 [cit. 2016-05-26]. Dostupné z: https://ext.eurocontrol.int/aixmwiki_public/bin/view/AIXM/Class_Airspace

[31] POROSNICU, Eduard, VEMBAR, Navin (ed.). *AIXM 5 - Temporality Model* [online]. 2010, , 32 [cit. 2016-05-26]. Dostupné z: http://www.aixm.aero/sites/aixm.aero/files/imce/AIXM51/aixm_temporality_1.0.pdf

[32] COWELL, Deborah a Eduard POROSNICU (eds.). *AIXM 5 - Feature Identification and Reference: use of xlink:href and UUID* [online]. 2011, , 18 [cit. 2016-05-26]. Dostupné z: http://www.aixm.aero/sites/aixm.aero/files/imce/AIXM51/aixm_feature_identification_and_reference-1.0.pdf

[33] Spatial Data Types Overview. *MSDN: Microsoft Developer Network* [online]. Microsoft Corporation, 2016 [cit. 2016-05-26]. Dostupné z: https://msdn.microsoft.com/en-us/library/bb964711.aspx

[34] ASP.NET Data Access Options: Object-Relational Mappers (ORM). *Microsoft Developer Network* [online]. Microsoft Corporation, 2016 [cit. 2016-05-26]. Dostupné z: https://msdn.microsoft.com/en-us/library/ms178359.aspx#orm

[35] Entity Framework. *Data Developer Center* [online]. Microsoft Corporation, 2015 [cit. 2016-05-26]. Dostupné z: https://msdn.microsoft.com/en-us/data/ef.aspx

[36] KRILL, Paul. Microsoft open-sources Entity Framework: Code release of .Net application development tool is being handled by Microsoft Open Technologies. INFOWORLD, INC. *InfoWorld, Inc* [online]. 2012, 2012-06-20 [cit. 2016-05-26]. Dostupné z: http://www.infoworld.com/article/2617690/microsoft-net/microsoft-open-sources-entity-framework.html

[37] Entity Framework. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-05-26]. Dostupné z: https://en.wikipedia.org/wiki/Entity_Framework

[38] MILLER, Rowan. *Entity Framework Development Workflows* [online]. [video]. Dostupné také z: https://msdn.microsoft.com/en-us/data/jj590134

[39] Supported elements. *Schemas.microsoft.com: version 3.1.1* [online]. Microsoft Corporation, 2016 [cit. 2016-05-26]. Dostupné z: http://schemas.microsoft.com/sqlserver/profiles/gml/SpatialGML.xsd

[40] VYMAZAL, Milan. *Publikace prostorových informací v kontextu EU/INSPIRE*. Brno, 2010. Dostupné také z: https://is.muni.cz/th/72855/fi_m/text_DP.txt. Diplomová práce. Masarykova univerzita Fakulta informatiky. Vedoucí práce RNDr. Milan Drášil, CSc.

[41] VINDEX. Problem converting to SqlGeometry from gml. In: *Microsoft Developer Network Forum* [online]. Microsoft Corporation, 2010 [cit. 2016-05-26]. Dostupné z: https://social.msdn.microsoft.com/Forums/sqlserver/en-US/2a2cea5c-196f-476e-aaeb-53be6c7ae39e/problem-converting-to-sqlgeometry-from-gml?forum=sqlspatial

[42] Factory (object-oriented programming). In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2016 [cit. 2016-05-26]. Dostupné z: https://en.wikipedia.org/wiki/Factory_(object-oriented_programming)

[43] WHITESIDE, Arliss (ed.). *Definition identifier URNs in OGC namespace: Version: 1.1.2* [online]. Open Geospatial Consortium Inc., 2007, , 46 [cit. 2016-05-26]. Dostupné z: https://portal.opengeospatial.org/files/?artifact_id=24045

# Seznam obrázků

Obrázek 1: Přehled vybraných interpretací vzájemných pozic (Dostupné z: [16]) ..... 12
Obrázek 2: Příklad křivky Curve složené z typů Arc a CubicSpline (dostupné z: [1]) ..... 15
Obrázek 3: Příklad konceptuálního modelu (dostupné z [27]) ..... 18
Obrázek 4: Přehled vývoje jednotlivých Property v čase (Dostupné z [31]) ..... 19
Obrázek 5: Vztah AIXMFeature a AIXMTimeSlice (Dostupné z [31]) ..... 20
Obrázek 6: Vývoj prvků Property v rámci prvků TimeSlice (Dostupné z: [31]) ..... 21
Obrázek 7: Diagram struktury geometrií v AIXM 5.1 ..... 24
Obrázek 8: Přehled geometrií v Microsoft SQL Server (dostupné z [33] ..... 25
Obrázek 9: Diagram základních entit ..... 30
Obrázek 10: Diagram řešení názvů Property přímo pod AIXMTimeSlice ..... 31
Obrázek 11: Diagram řešení názvu Property v rámci pole v AIXMProperty ..... 31
Obrázek 12: Diagram všech databázových entit ..... 33
Obrázek 13: Přehled struktury objektu Linq To XML ..... 34
Obrázek 14: Přehled možných přístupů k databázi (dostupné z: [38]) ..... 35
Obrázek 15: Diagram třídy pro parsování dat ..... 37
Obrázek 16: Ukázka konzumace souboru jednotlivými metodami parseru ..... 38
Obrázek 17: Diagram třídy pro tvorbu prvků Feature z AIXM 5.1 ..... 38
Obrázek 18: Diagram třídy pro přístup do databáze ..... 39
Obrázek 19: Diagram celkového pohledu na návrh modulu ..... 40
Obrázek 20: Diagram implementace třídy pro načítání dat ..... 41
Obrázek 21: Diagram celkového pohledu na modul ..... 42

# Seznam tabulek

Tabulka 1: Přehled podobností datových typů SQL Serveru Spatial a geometrií GML ..... 27

# Seznam kódů

Výpis 1 Příklad formátu GeoJSON (dostupné z: [18] ) ..... 12
Výpis 2: Příklad deklarace atributu srsName ..... 16
Výpis 3: Příklad lokální reference v AIMX 5.1 ..... 22
Výpis 4: Příklad externí reference v AIXM 5.1 ..... 22
Výpis 5: Příklad abstraktní reference v AIXM 5.1 ..... 23
Výpis 6: Příkaz ke zjištění dostupných projekčních systémů ..... 33
Výpis 7: Ukázka příkazu pro parsování GML geometrií ..... 36
Výpis 8: Ukázka příkazu pro zpětný výpis do GML ..... 36

# Obsah CD

| | |
|---|---|
| Reprezentace_dat_AIXM_5.1_v_relační_databázi.pdf | Dokument této práce |
| /diagramy | Složka obsahující vybrané diagramy v plné velikosti |
| /implementace | Složka obsahující implementaci modulu |
| /testData | Složka s dodanými testovacími daty |