T.C.
ÇEVRE VE ŞEHİRCİLİK
BAKANLIĞI

COĞRAFİ BİLGİ SİSTEMLERİ GENEL MÜDÜRLÜĞÜ

TÜRKİYE KENT BİLGİ SİSTEMİ STANDARTLARININ BELİRLENMESİ PROJESİ:

KAVRAMSAL MODEL BİLEŞENLERİ

3- METODOLOJİ

TÜRKSAT

Yüklenici

İSTANBUL TEKNİK ÜNİVERSİTESİ 1773

Alt yüklenici

Temmuz 2012

İçindekiler

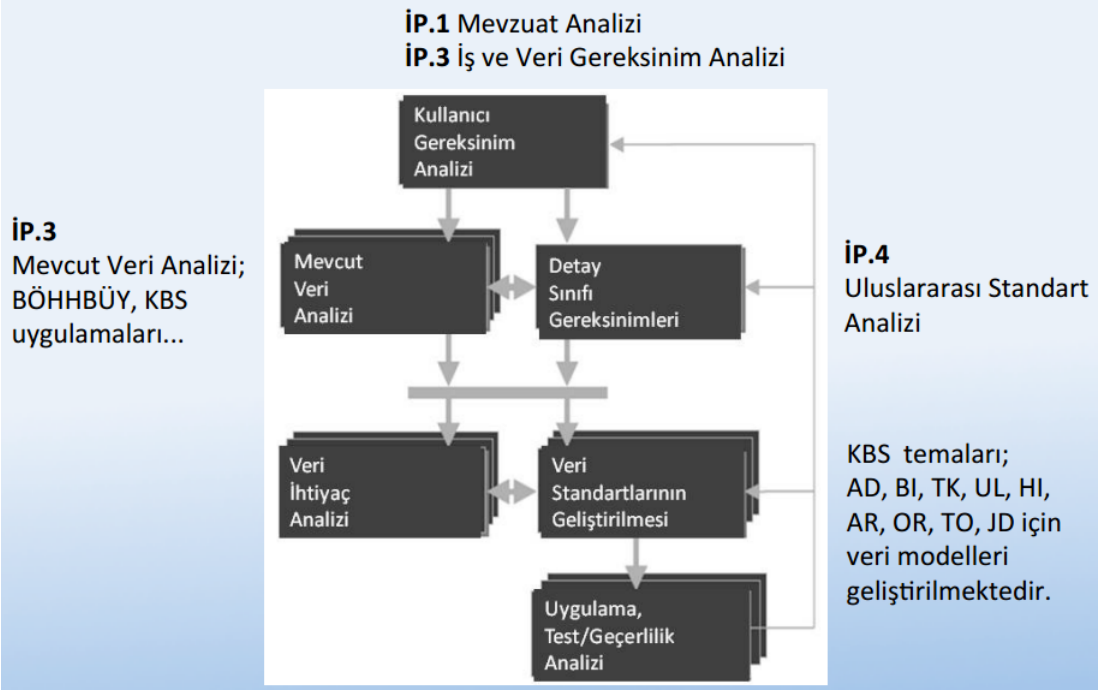
Şekiller Listesi

Tablolar Listesi

1 Metodoloji

1.1 Belirtim Geliştirmenin İlkeleri

Kavramsal çerçevenin bir parçası olarak belirtim geliştirme metodolojisi; işlemlerin devam etmesi ve KVA içindeki tekrar kullanım, fizibilite ve orantısallık konularının devamı açısından önem taşımaktadır. Metodoloji, işlemin hangi evresinde hangi adım atılacağı konusunda talimat verir.



Şekil 1 KBS Standart Geliştirme Metodolojisi

Şekil 1'de KBS standart geliştirme metodolojisi gösterilmiştir.

Belirtim geliştirme işlemi veri sağlayıcıları ve/veya veri kullanıcıları tarafından yürütülür. Sağlayıcı yaklaşımına göre asıl amaç özel bir temaya ait mevcut veri setleri arasında ortak bir payda bulmaktır. Bu yaklaşımda dışardan bir gösterge olmadığında, birlikte çalışabilirlik gereksinimleri belirsiz kalabilmekte ve bu da aşağıdaki problemlere yol açabilmektedir;

- Birlikte çalışabilirlik esaslarına göre hazırlanmış veri kullanıcının gereksinimlerini karşılamıyor olabilir.

- Birlikte çalışabilirlik için optimum seviyeyi yakalamak yerine, en güçlü paydaşlar ürettikleri veri setinin potansiyel dönüşümlerini/değişimlerini minimum seviyeye indirmek için kendi çözümlerini destekleyebilirler.

Kullanıcı yaklaşımına göre dışardan alınan göstergeler kullanıcı gereksiniminden doğar ve belirtim geliştirme işleminin başında dikkatlice analiz edilmiş ve şekil verilmiştir. Bu yaklaşımın riskleri aşağıda sıralanmıştır;

- Kullanıcı, gereksinimlerini detaylı şekilde öne çıkarmak çok zordur
- İfade edilen gereksinimler çok iddialı olabilir. Bu da fazla masrafa ve eldeki verilerin uygulamaya konmasının imkansızlığına yol açabilir
- Tekrar kullanıma odaklanmak yerine, belirtim işlemi "güçlü" kullanıcının ihtiyaçlarını karşılayacak ürün elde edebilmektedir

Her iki yaklaşımın kombinasyonu ile teknik ve finansal birliktelik dengelenebilir. İstekler ile uygulanabilirlik arasında bir denge bulunmalıdır. İsteklerin çok fazla olması uygulanması zor ve aşırı masraflı karmaşık belirtimlere yol açabilmektedir. Ayrıca belirtimler çok karmaşık olursa veri sağlayıcılar tarafından desteklenmeme ve kullanıcılar tarafından benimsenmeme riskini barındırmaktadır. Diğer taraftan çok basit veri belirtimleri yetersiz birlikte çalışabilirliğe, harcanan emeklerin boşa gitmesine ve altyapı yararlarının somut hale getirilememesine yol açabilir. Çözülmesi gereken ana noktalar Şekil 2'de gösterilmiştir.

Şekil 2- Veri belirtim işleminde denge yerini bulma

Dengeyi bulabilmek için iki ana yol bulunmaktadır;

- Odak noktası, daha geniş kullanım için tutarlı konumsal (zamansal) bilgi üzerinde olmalıdır.
- Model için genişletme mekanizması olmalıdır ve diğer konumsal ve konumsal olmayan öğeler modele nasıl bağlanır gösterilmelidir.

1.2 Veri Belirtimi Gelişim Döngüsü

1.2.1 "Use case" toplama ve geliştirme

Bir "use case" amaç odaklı olarak sistem ile aktörler arasındaki etkileşim ifade edilir. Use case, kullanıcının gereksinimlerini anlamaya ve bunları sağlamak için veri tanımlamaya yardımcı olur.

Bir use case birkaç veri temasını kapsayabilir. Örneğin belli bir alandaki sel riski analizini anlatan use case hidrografya, yükseklik, meteoroloji verilerine de ihtiyaç duyabilir ve "doğal risk alanları" veri teması için bir girdi oluşturabilir. Ortak use case nihai çapraz tema bağımlılıklarını açıklamaya yardım eder. Bu yüzden use case veri kullanımının katlılığını yansıtmak zorundadır.

Gereksinimlerin doğru değerlendirilmesi açısından, use case'lar öncelik sıralamasına göre sıralanmalıdır. Yüksek öncelik çok kullanıcı senaryolarına veya kritik zamanlı (afet yönetimi, sel vb.) senaryolara verilmelidir.

Pratikte paydaşlardan use case toplamak zor olabilir. Veri kullanıcıları KVA'nın faydalarından fazla haberdar olmayabilirler. Bu durum belirtim geliştirme işlemini tehlikeye sokmamalıdır.

1.2.2 Kullanıcı Gereksinimlerinin ve Konumsal Nesne Tiplerinin Belirlenmesi

Veri modelinin ilk kısımlarında use case konumsal veri gereksinimlerini belirlemek için kullanılır. Bu model, konumsal nesne tiplerinden, taslak tanımlarından ve açıklamalarından ve diğer veri belirtim elemanlarından aday listesi içerir. Bu elemanların herbiri detay seviyesine göre tanımlanır. Konumsal

nesne tiplerinin kavramları paylaşılmalı ve farklı temalarla uyumlu hale getirilmelidir. Bu içerik için faydalı bir araç Feature Concept Dictionary36'dır.

1.2.2.1 As-is analizi

KVA mevcut veriyi biraraya getirmelidir, use case'lerdeki veri gereksinimleri mevcut "as-is" durumuyla karşılaştırılmalıdır. Bu analiz istenen verinin, veri sağlayıcılar tarafından sağlanıp sağlanamayacağını ortaya çıkarmaktadır. Eğer sağlanırsa dönüşüm işindeki karmaşıklığı da göstermiş olacaktır. "as-is" analizi genellikle gap analiziyle paralel yürümektedir.

1.2.2.2 Gap analizi

Gap analizi mevcut veri ile karşılanamayan kullanıcı gereksinimlerini belirler. İki çeşit boşluk vardır. Teknik boşluk herhangi bir veri setinden yada veri dönüşümünden toplanan veriyle doldurulabilir. Diğer taraftan içerik boşluğu sadece veri toplamayla halledilebilir.

Teknik boşlukları doldurmak kullanıcı için bir artıdır fakat veri sağlayıcıya önemli bir maliyet getirir. Veri entegrasyonu ve dönüşümü sağlayan otomatik araçlar yardımıyla sorun çözülebilir fakat bu araçlar her zaman mevcut olmayabilir. Bu yüzden fayda ile maliyet karşılaştırması yapmak gerekmektedir.

1.2.3 Veri Belirtimi Geliştirme

Başlangıçtaki veri modeli ve diğer veri belirtim elemanları as-is ve gap analizi sonuçlarına göre ayarlanmalıdır. Teknik ve finansal fizibiliteye göre veri belirtimlerinin içeriği zorunlu veya opsiyonel olarak ayrılabilmelidir.

Çok önemli yada az önemli konumsal nesne tiplerini ayırmak gerekiyorsa, her iki grup ayrı veri modellerinde paketlenmelidir. Ana gereksinimleri destekleyen konumsal nesneler çekirdek modele yerleştirilmiştir.

Teknik elemanların yanında veri belirtimleri daha iyi anlama ve uygulama açısından açıklamalar ve örnekler de içerebilir.

1.2.4 Uygulama, Test Etme ve Doğrulama

Belirtimler paydaşlar tarafından gözden geçirilmelidir. Böylece veri belirtimlerinin altyapı amacına uygun olup olmadığı, uygulamayı destekleyen yeterli veri içerip içermediği sağlanmış olur.

Belirtim testi, kullanım için fizibilite veya uygunluk geri bildirimi sağlamak için yürütülebilir. Fizibilite, veri sağlayıcıların verilerini birlikte çalışabilirlik amacına uygun olarak dönüştürme çalışmalarını içermektedir. Sonuç olarak da teknik fizibilite ve uygulama maliyetleri konularında geri dönüşümü sağlanmış olmaktadır.

Uygulama testleri; kullanıcıların işlerinin birlikte çalışabilirlik sayesinde ne kadar kolaylaştığını içermektedir. Veri kullanıcıları tarafından yapılan bu test, sağladıkları verinin performanslarını içermektedir.

Test ve paydaş danışmanın sonuçları herhangi bir adımda veri belirtim işlemini tekrar belirtmek için kullanılabilmektedir. Bu tekrar belirtimler bir anlaşmaya varılana kadar devam etmektedir. Doğrulama işleminden sonra belirtimler kamu tarafından kullanılmak üzere basılmaktadır.

Yasal dayanağı olan KVA'lar için gerekli olan bir adım daha vardır. Teknik içeriği korurken yasal gereksinimleri sağlamak için teknik taslak hazırlanmalıdır. Yasal mevzuatı sağlamak için bir yol altyapıdaki mevcut veriler yoluyla servislerdeki sadece parametreleri kullanmaktır. Diğer bir yol ise teknik fizibilite ve maliyet konusuna dayanan veri belirtimlerinin bir altkümesini seçmek, semantik modeli içermektir. Bu durumda tam teknik içerikli veri belirtimleri paydaşlar için bir rehber özelliği taşımaktadır.

1.3 Basisregistratie Grootschalige Topografie(Büyük Ölçekli Topografya Kayıtlaması)

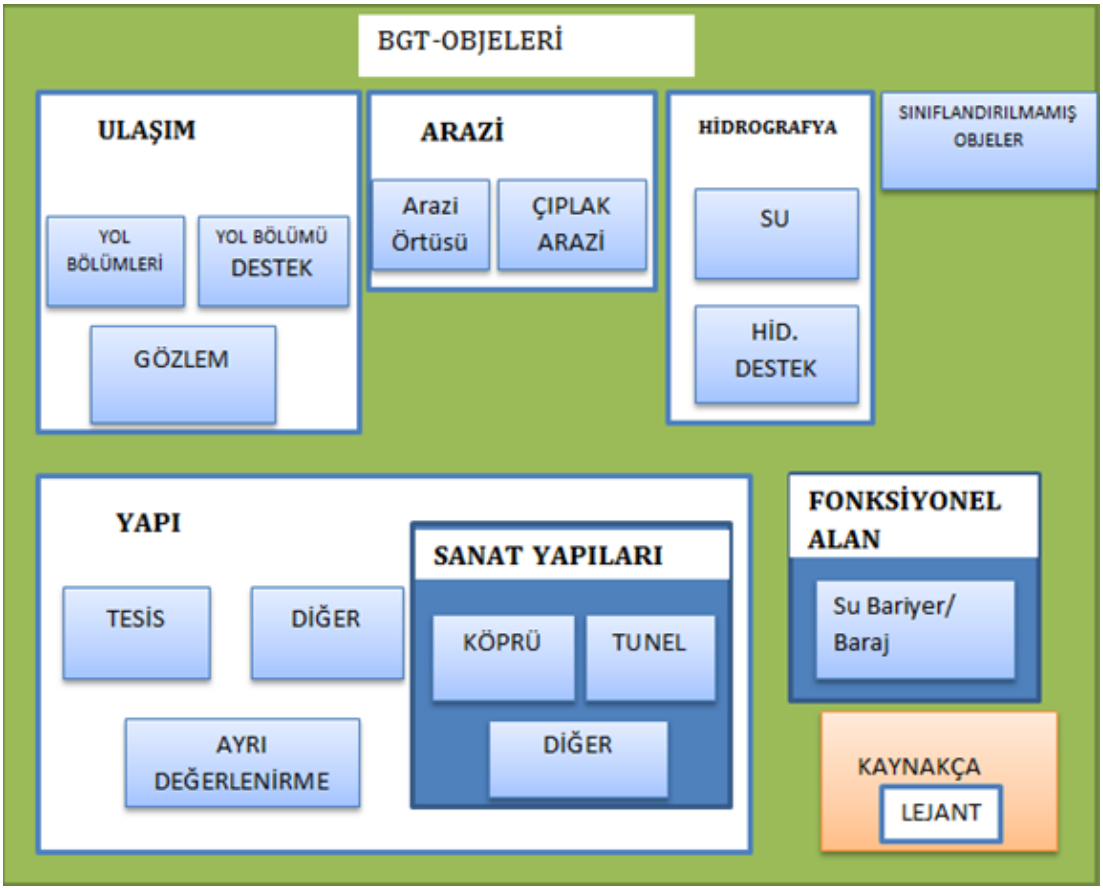
Ulusal coğrafi veri standardizasyonu ve kayıt altına alınması projesi kapsamında 2007 yılında IMGeo, versiyon 1.0 olarak uygulamaya geçmiştir. Bununla beraber 2011 yılına gelindiğinde tasarımı tamamlanan BGT projesi, 2012 yılında nihai

kararla yürürlüğe girerek başta Geonowum olmak üzere birden çok kuruluşun birlikte çalışarak geliştirdiği veri kataloğunu ortaya çıkarmıştır. İki yıllık test süreci boyunca BGT ile uyumlu IMGEO modeli tasarlanmış, 3D topografik modelleme oluşturulmuştur. IMGeo'nun 2.0 sürümü kapsamında BGT'nin yapmış olduğu çalışmaların tamamı kapsanmış ve detay alım çalışmaları ayrıntılı olarak incelenmiştir. IMGeo, coğrafi veri kayıt sistemleri için birincil proje olmakla birlikte; BGT, yönetim planları ve ilave topografik içerik çalışmaları IMGeo projesine entegre edilerek opsiyonel olarak kullanıcıya sunulmuştur.

BGTIMGEO Modelin içeriği;

1. Veri Kataloğu
2. Nesne Bilgisi İçeren Kitapçık
3. Görselleştirme
4. Standardizasyon

BGT nesnelerini özetleyen ve gruplandıran şekil ve diyagramlar Şekil 3'te belirtilmiştir.



Şekil 3 BGT Objeleri

Ana ve alt gruplar altında nesne sınıflandırmaları yapılmıştır. Meta bilgiye ilişkin veri kalite ve kaynak bilgileri tanımalanabilir bir geometrik ifade içerisinde mevcuttur.

BGT, Hollanda'nın topografik altyapısını belirli bir içerik ve standartta nesne kayıtlaması çalışmasıdır. Topografik haritalar bütünlük, doğruluk ve güncel veri çatısı altında 1:500-1:5000 ölçek aralığında düzenlenmiştir. Yollar, binalar, demiryolları, akarsular, parklar, ormanlar ve bunlar gibi detay sınıfları topografik harita altlığı üzerinde konumlarına göre konumlandırılmıştır. BGT kullanım kataloğu yardımı ile BGT kullanıcısı çalışma konularına bağlı olarak bu portaldan ihtiyaç duydukları veri gruplarını alabilir; ürettiği verileri de aynı zaman paylaşıma sunabilir.

2 Uygulama Şeması ve Detay Katalogu Üretim Kuralları

Coğrafi veri yönetimine yönelik kavramsal şemanın oluşturulması için UML (Unified Modelling Language) olarak ifade edilen Tekil Modelleme Dili kullanılmıştır. TRKBİS kapsamında TUCBS Projesinde kullanılan ve ISO 19103'de tanımlanan kavramsal şema kuralları benimsenmiştir (ISO/TC211, 2005b)

UML, ISO/IEC PAS 19501-1 ve UML notasyon rehberinde anlatıldığı şekilde UML 1.3 ile tutarlı bir şekilde olmalıdır. Normatif modeller olarak sınıf diyagramı (*class diyagram*) ve paket diyagramlar (*package diyagram*) kullanmalıdır. Diğer UML model diyagram tipleri de bilgi amaçlı kullanılabilir. Bütün normatif modeller; öznitelikler (*attributes*), ilişkiler (*associations*), işlemler (*operations*) ve veri tipi tanımları için tanımlar içermektedir.

Her bir veri teması için nesneye yönelik; ilişkisel UML uygulama şemaları ve detay katalogları üretilmektedir. Tablo 1'de stereotype'lar ve kullanılan standartlar gösterilmiştir. TRKBİS Uygulama şeması oluşturulmasında, ISO 19109 ve TUCBS Kavramsal Model temel alınmıştır. TRKBİS Detay Tipi tanımlanırken ISO 19136 ve TUCBS geometri standartları kullanılmıştır. Veri tipi tanımlanırken ISO 19103, kod listesi ve öznitelik değeri tanımlanırken ISO 19136 temel alınmıştır.

Tablo 1 Stereotype ve Kullanılan Standartlar

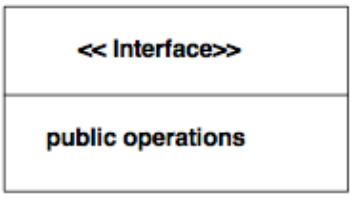
Stereotype	**Kullanılan Standart**
Uygulama Şeması (Application Schema)	ISO 19109 ve TUCBS Kavramsal Modele Göre
Detay Tipi (Feature Type)	ISO 19136 ve TUCBS Geometri'ye Göre
Veri Tipi (Data Type)	ISO 19103
Kod Listesi (Code List)	ISO 19136
Öznitelik Değeri (Enumeration)	ISO 19136

2.1 Sınıflar (Classes)

Sınıflar; aynı öznitelik, fonksiyon, metod, ilişki, davranış ve kısıtlamaları içeren nesneler kümesini ifade eder. Sınıf, modellenmek istenen bir kavramı ifade eder; platformdan bağımsız veya platforma özgü olabilir. UML sınıfı; bir isme, özniteliklere ve ilişkilere sahiptir. İhtiyaca göre fonksiyonları ve kısıtlamaları tanımlanabilir.

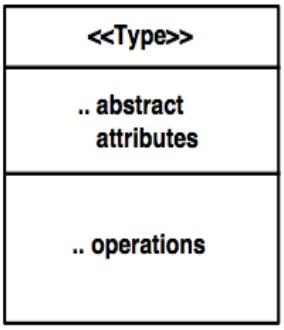
UML stereotype olarak <<interface>> ve <<type>> gibi iki farklı sınıf tipini sahiptir.

Şekil 4'deki <<interface>>, bir nesnenin davranışlarını karakterize eden işlemlerdir ve öznitelik içermezler.



Şekil 4 Stereotype<<interface>>

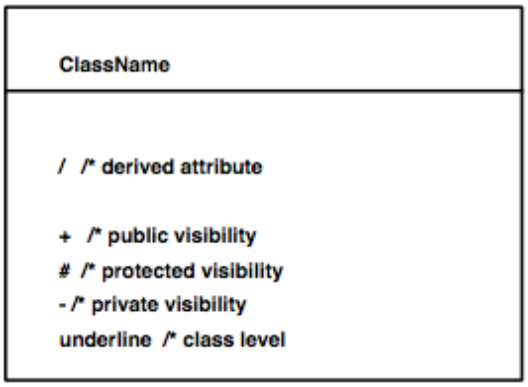
Şekil 4'teki <<type>>, bir stereotype sınıftır ve içinde nesnelere uygulanabilen işlevler/fonksiyonlar, öznitelikleri ve ilişkileri içerir.



Şekil 5 Stereotype <<type>>

Öznitelik ve işlemlerin görünebilirliği Şekil 6'te ifade edilmiştir. "Public" belirtilmesi genel kullanım için uygun iken, "Protected" ve "private" belirtilmesi

gizli ve özel kullanım için standart belirtimlerde kullanılmamaktadır. Uygun görünebilirlik sembolleri kullanılmalıdır.



Şekil 6 Görünebilirlik Sembolleri

2.2 Öznitelikler (Attributes)

Öznitelikler için UML notasyonu aşağıdaki gibidir;

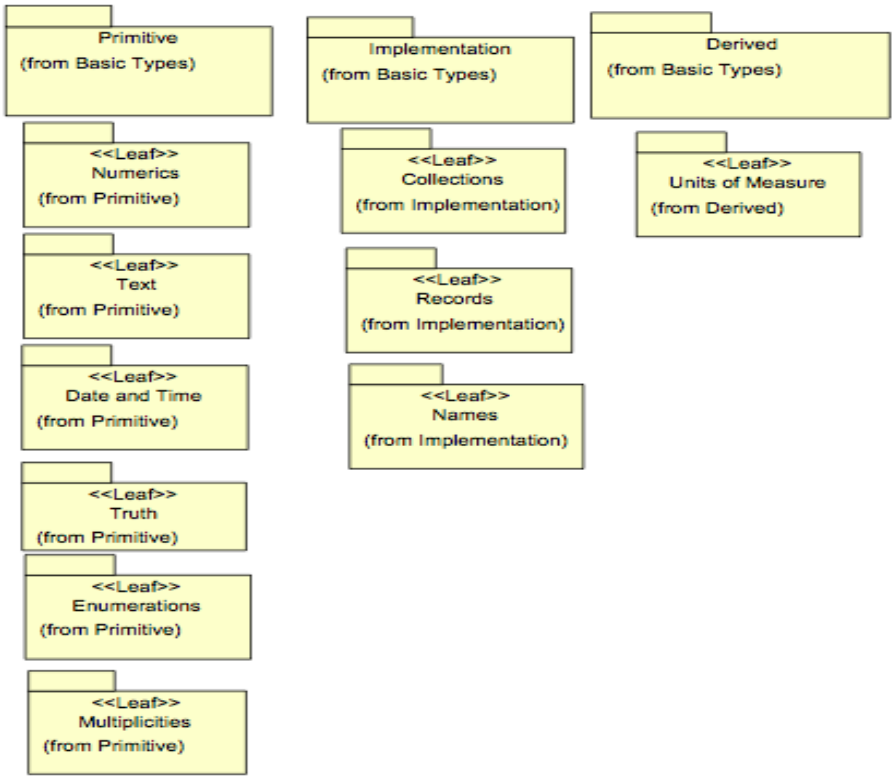
<<stereotype>> [görünebilirlik] name [multiplicity] [:type] [= initial value] [{property-string}]

Öznitelikler sınıf içine tek tanımlanmış olmalıdır. Öznitelik türetilmiş ise soyut öznitelik gibi davranır ve türetilmesine bağlı olarak kurallara göre çağrılır. Bütün öznitelikler bir tipe sahip olmalıdır. Öznitelik tiplerinin varsayılanı olmadığından mutlaka tip tanımlaması yapılmalıdır. Belli bir tekrarlılık değeri verilmemişse çokluk (1) olarak kabul edilir.

2.3 Temel Veri Tipleri (Basic data types)

Temel tipler Şekil 7'te görüldüğü gibi üç tipte toplanmıştır:

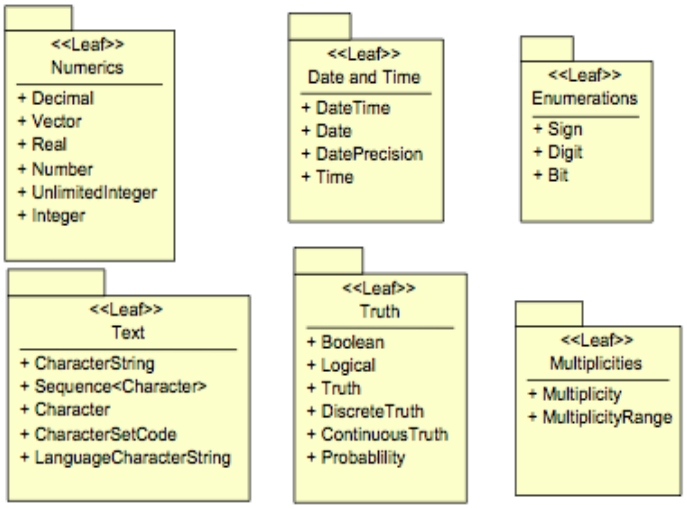
1. **Basit tipler:** Değerlerin sunumu için temel tiplerdir. Örnek olarak; CharacterString, Integer, Boolean, Date ve Time, verilebilir.
2. **Uygulama tipleri:** Değer tiplerinin sıklıklarını gösterir. Örneğin Set, Bag, Sequence ve Record.
3. **Elde edilen tipler:** Ölçüm tipleri ve ölçüm birimleri.



Şekil 7 Temel Tipler

2.3.1 Basit tipler (Primitive types)

Şekil 8'da basit tiplerin genel görünümü sunulmuştur.



Şekil 8 Basit Tipler

2.3.1.1 Numerik Tipler

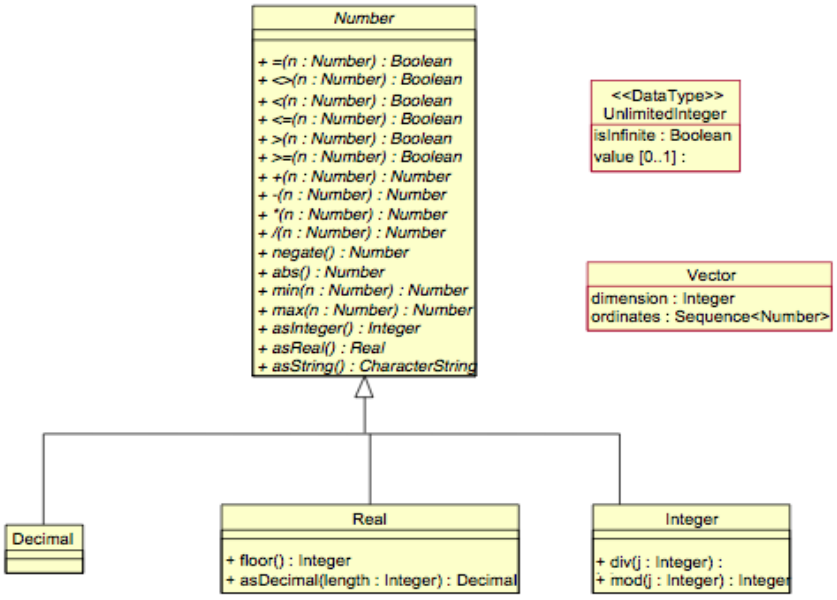
Şekil 9'de nümerik tipler gösterilmiştir.

Integer: Tamsayıları ifade etmede kullanılır. Örnek; 12, -24784

Decimal: Tam ondalığa sahip veri tipidir. "Float" ifade yaklaşık değer verir iken "decimal" tam değer verivermektedir. Örnek 12.75

Real: Bir tam kısım bir de üslü kısımdan oluşan işaretli reel sayıdır. Uzunluğu kullanıma bağlıdır. Örnek 23.501, -1.234E-4, -23.0

Vektör: Koordinat sistemindeki yeri gösteren koordinatlı sıralı sayılardır. Bir koordinat herhangi boyutta herhangi bir sayı tipi olabilir. Örnek (135,560,143).



Şekil 9 Nümerik Tipler

2.3.1.2 Yazı tipleri

Character String: Karakter kümesinden seçilen ve özel karakterler de içeren rasgele uzunlukta karakter dizisidir. Aşağıdaki karakter kümelerinden birini kullanır:

ISO/IEC 10646-1: Universal Character Set (UCS), ISO 10646'nın temel çok dilli uygulama modelidir.

ISO/IEC 10646-2: Universal Character Set UCS-4

Örnek "Ærlige Kåre så snø for første gang."

2.3.1.3 Tarih ve Zaman

Date: Tarih yıl, ay ve gün biçiminde ifade edilir. Karakter kodları string biçimindedir ve ISO 8601'de belirtilen tarih formatına uygun biçimde tanımlanır. Örnek; 1979-03-26

Time: Zaman saat, dakika ve saniye biçiminde ifade edilir. Karakter kodları string tipindedir ve ISO 8601de belirtilen tarih formatına uygun biçimde olmalıdır. Zaman dilimi bulunması seçimliktir. Örnek: 17:47:34 veya 17:47:34+01:00

Date and Time: Tarih ve zamanın kombinasyonudur. Karakter kodlaması ISO 8601'e uygun olmalıdır.

Boolean: Doğru/Yanlış, Var/Yok gibi ifade edilen değerlerdir.

2.3.1.4 Öznitelik değer tipleri (Enumeration) ve kod listesi (Code lists)

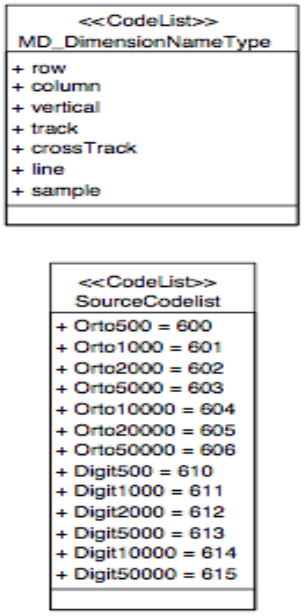
Özniteliğin alabileceği değer tiplerini ifade eder. İlgili öznitelikler sadece bu değer listesinden değer alır. Örneğin bina tipi özniteliği; Enum Building Type{Public,private,tourist} olarak tanımlanır.

Kullanıcı tanımlı veri tipidir. Değer tiplerinde genişleme olursa şema değişikliği gereklidir. Bu yüzden değer tiplerinin kesin olduğu ve genişlemenin olmadığı durumlarda öznitelik değer tipleri kullanılır. Eğer değişiklik bekleniyorsa kod listesinin kullanılması uygundur.

Kod listesi, karakter değerleri kullanan esnek bir öznitelik değer tipidir. Öznitelik değer kod listesi ISO 3166-1 standartlarına göre kodlanabilir. Kod

listeler kullanıcıya değerlerini gösteren yapıdadır ve bu yüzden de mnemoniktirler. Kod listesinin seçilmiş gösterimi değer/kod şeklindedir.

<<CodeList>> başlangıç değeri olarak bir öznitelik ismi ve bir koddan oluşur. Öznitelik isminin bulunduğu durumlarda kod listesi ile eşdeğerdir (Şekil 10).



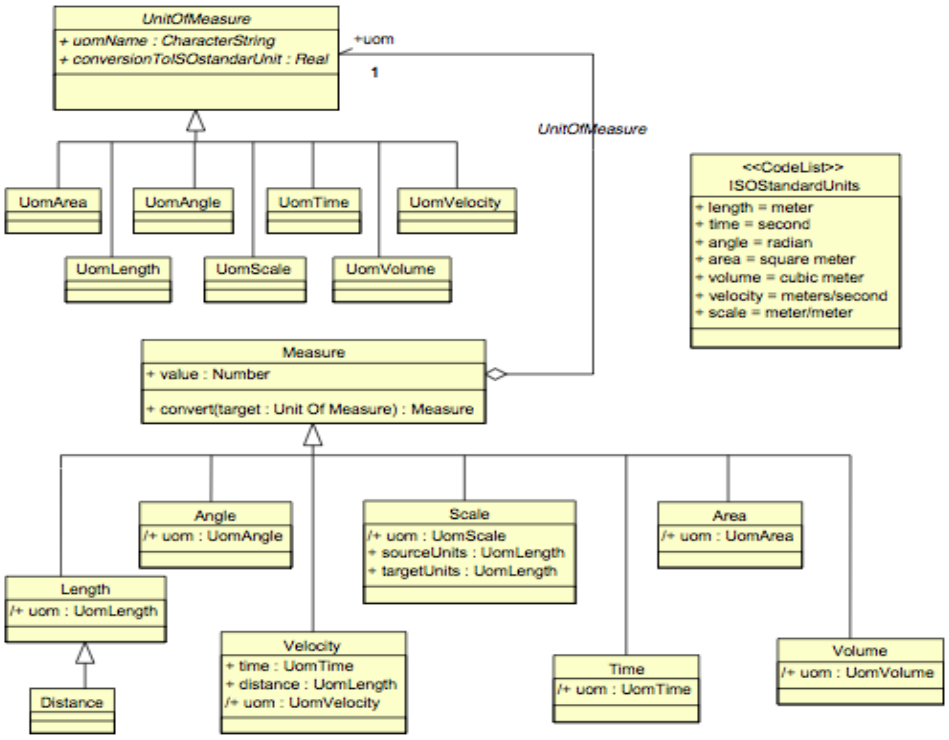
Şekil 10 CodeList Örnekleri

Tip ve öznitelik isimleri için jenerik ve yerel yapıda isimler üretmek için kullanılır.

2.3.2 Elde edilen tipler

Measure: Bazı varlıkların boyut, miktar ve genişliklerinin elde edilmesiyle çıkarılan sonuçtur.

UnitOfMeasure: mesafe ve alan gibi fiziksel nicelikleri ölçmede kullanılan birimdir.



Şekil 11 Ölçüm birimi Tipleri

2.3.2.1 "Null ve "Empty"

Null istenen değer tanımsız anlamına gelir. Bütün Null değerler eşdeğerdir. Bir nesnenin istendiğinde "Null" değerine dönmesi, belirtilen kriterleri karşılamadığını gösterir. "Empty" ifadesi ise içinde eleman bulunmayan kümeleri ifade eder. Başka bir ifadeyle, matematikte olduğu gibi boş küme olduğunda kullanılmaktadır.

2.3.2.2 İşlemler (Operations)

Koşula bağlı öznitelik tanımlamaları ve modeldeki işlemler ve fonksiyonlar UML gösterimi şöyledir:

<<stereotype>> [visibility] name [(parameter-list)] [:return-type] [{property-string}] [(parameter-list element)] ::=

[direction] name : type [= default-value]

2.3.2.3 İlişkiler (Associations)

İlişki: iki sınıf arasındaki anlamsal ilişkiyi ifade eder.

Kalıtım: Alt detay tipi, üst sınıfın tüm özelliklerine sahip türeyen bir alt sınıftır. Örneğin; Belediye detay sınıfından (Sınıf A), İlçe (Sınıf B) ve Belde (Sınıf C) belediyesi gibi alt detay sınıfları türer.

Bir elemanın diğeri tarafından kullanılmasıdır.

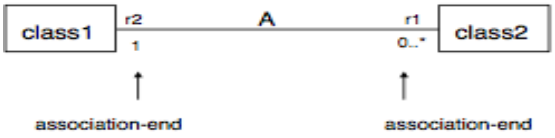
Soyutlama düzeyindeki değişimi ifade eder.

Bütünleme: Karmaşık ilişkileri daha basit parçalara ayırmada kullanılabilir. Böylelikle bir sınıf basit alt parçalara bütünleşmiş hali olarak düşünülebilir.

Oluşum: Bir veya daha fazla detay sınıfı oluşum sınıfının parçasıdır. Oluşum sınıfı var olmadığında alt diğer sınıflar da var olamaz.

İki veya daha fazla sınıflar (class, interface, type, vb.) arasındaki anlamsal bağlantıya ilişki (association) denir. İki veya daha fazla sınıf arasındaki ilişikiyi tanımlamak için de kullanılabilir. Bunun dışında, oluşum (composition) ve bütünleme (aggregation) olarak ifade edilen iki ayrı özel tip ilişki kullanılabilir. Her üç tip farklı anlam taşımaktadır. İlişki ile iki sınıf arasındaki genel ilişki tanımlanıyor iken, oluşum ve bütünleme ilişkisi parça-bütün ilişkisinde kullanılır.

İkili ilişki, bir isme ve iki bağlantı ucuna sahiptir. Bağlantı ucunun rol ismi, kardinallik ifadesi ve uygulama ihtiyacına göre bütünleme ve oluşum gibi sembolleri mevcuttur. Bağlantı ucu da daima bir sınıfa bağlanmalıdır.

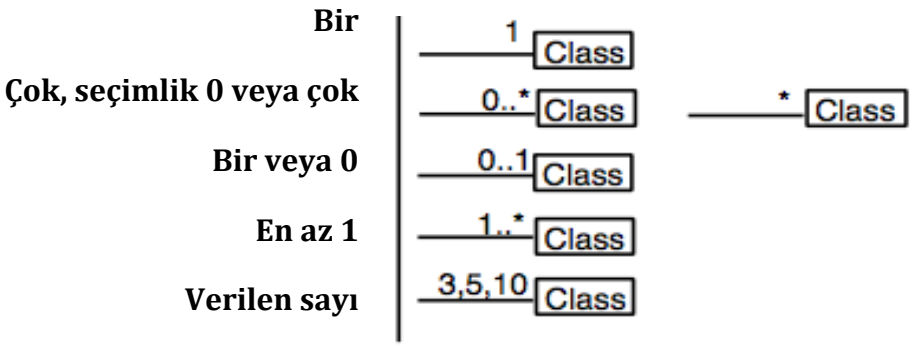


Şekil 12 İlişki tanımlama

Şekil 12'da A isimli ilişki iki ilişki ucuna sahip olup, rol ismi ilişkinin sonunu belirlemek için kullanmaktadır. Rol ismi *r1 class2* isimli sınıfa bağlanan ilişki ucunu gösterir. İlişki ucunun çokluğu sadece bir(1), sıfır veya bir (0..1), bir veya çok (1..*), sıfır veya çok (0..*) ya da aralık (m,n) seçeneklerinden biri olmalıdır.

Belirli bir sınıftan karşı ilişki ucundaki rol ismi hedef sınıfın rolünü belirlemektedir. Class2, class1'e sadece bir çoklukla ve r2 rolüyle bağlıdır diyebiliriz. Diğer şekliyle class1, class2'ye r1 rolüyle ve sıfır veya çok/sonsuz çokluğuyla bağlıdır.

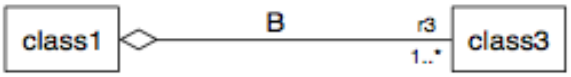
Örnek ilişki tipleri olarak, Şekil 13'deki gibi; 0..1 (sıfır veya bir), 0..* (sıfır veya daha fazla), 1 (bir), 1..* (en az 1), 2..5 (iki … beş) vb. çokluk tanımlanabilir.



Şekil 13 Çokluk Belirtimleri

Bütünleme, iki sınıftan birinin kapsayan diğerinin kapsanan olduğu bir ilişkidir. **Error! Reference source not found.** bir bütünleme örneğidir. Burada class1'e yakın olan uçtaki bütünleme sembolü class1'in class3'ten oluşan bir bütünleme olduğunu gösterir. Bunun anlamı class3 class1'in bir parçasıdır. Modelde class3

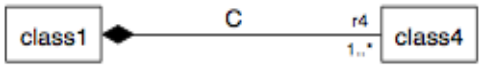
nesneleri bir veya daha çok class1 nesnelerini kapsar demektir. Kapsanan nesneler, kapsayan nesnelerin olmadığı durumlarda da varolabilir.



Şekil 14 Bütünleme (Aggregation)

Oluşum, güçlü bir bütünleme şeklidir. Burada kapsayan nesne silinirse kapsadığı bütün nesneler de silinir. Kapsayan nesneyi oluşturan parçalar kapsayan nesne olmadan varolamazsa composition kullanılır. Şekil 15'deki içi dolu elmas oluşum ilişkiyi ifade eder. Burada class1 nesneleri bir veya daha çok class4 nesnelerinden oluşur ve class1 objeleri olmadan class4 objeleri de olamaz. Kapsayan sınıf için kardinallik her zaman 1'dir. Kapsanan nesneler veya parçalar birkaç kapsayan arasında paylaşılamaz.

Bütün ilişkiler, ilişki uçlarında kardinaliteye sahip olmak zorundadır. Çeşitlilik her zaman en çok bir olmak zorundadır. En az bir rol ismi tanımlanmalıdır. Diğeri tanımlanmadığı sürece varsayılan olarak inv_rolename olarak alınır.



Şekil 15 Oluşum (Composition)

İlişkinin yönünü belirten ilişki uçları (roller) isimlendirilmelidir. Bu yapılmasa da en azından ilişkinin kendisine isim verilmelidir. İlişki uçları sınıf içinde tek olmak zorundadır. İlişkinin yönü belirtilmelidir. Eğer yön belirtilmemiş ise çift yönlü olarak kabul edilir. Tek yönlü ilişkiler için çizginin ucuna yön oku konulabilir. Sadece ilişkiye isim verilmişse ilişkinin yönünün belirtilmesi gerekir.

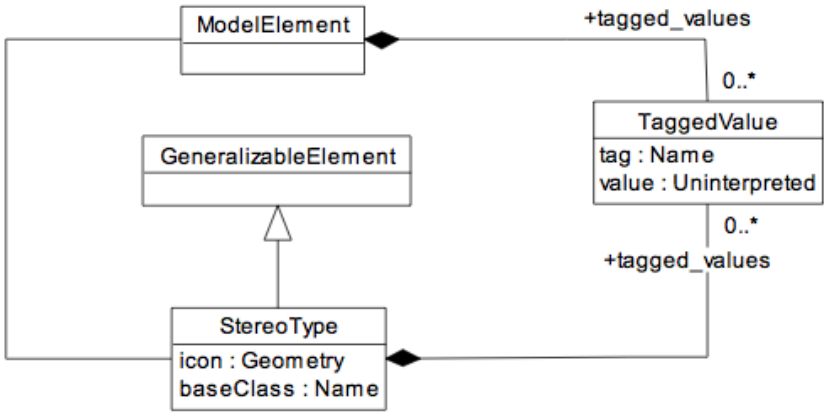
2.3.2.4 Stereotipler ve taglı değerler

Coğrafi bilgi ve servislerin modellemesinde UML kullanımını özelleştirmek için UML üç tane geliştirilebilir mekanizma kullanmaktadır. UML kavramları geliştirilebilir mekanizmalardan biri olan UML stereotipi kullanır. Diğer model elemanlarını sınıflandıran bir model elemanı da diyebiliriz.

Taglı değerler, UML için diğer bir geliştirme mekanizmasıdır. Taglı değer, UML'de herhangi bir model elemanına özellik eklemek için kullanılan tag değer çiftidir.

Taglı değerler, bir elemandaki isim (tag) ayıracı (=) özellikleridir. Teknik belirtimlerde aşağıdaki stereotipler kullanılmaktadır;

1) <<Interface>>, nesneler tarafından desteklenen işlem kümelerinin tanımıdır.
2) <<Type>>, nesnelerin domain belirtimi için kullanılan stereotipi sınıfı olarak öznitelikler ve ilişkilere sahip olabilirler.
3) <<Control>>, esas amacı servis sağlamak olan ve içindeki veriyi göstermeyen sınıf için yapılan UML tanımlamasıdır.
4) <<Entity>> bilgi taşıyan sınıfı ifade eder.



Şekil 16 UML Stereotipleri ve taglı değerler

2.3.2.5 Öznitelikler ve ilişkilerde seçimlik, koşula bağlılık ve zorunluluklar

UML'de bütün öznitelikler varsayılan tanımlandığında zorunludur. Öznitelik ve ilişki rol isimlerinde çokluk belirtilmesi ile seçimlik ve koşula bağlı öznitelikler tanımlanabilir. Eğer çokluk 0..1 veya 0..* ise öznitelik mevcuttur veya ihmal edilebilir.

Koşula bağlı öznitelik, OCL'de kısıtlayıcı ifadeye sahip seçimlik öznitelik olarak gösterilmelidir. Özniteliğin koşullu tanımlanması diğer özniteliklere bağlı olarak seçimlilik anlamına gelir.

Çokluk belirtilmemişse ilişki için varsayılan çokluk 0..* ve öznitelik için 1'dir.

2.3.2.6 İsimlendirme (Naming)

Bütün sınıflar tek isme sahiptir ve bir paketin içinde tanımlanmalıdır. Sınıf isimleri büyük harfle başlamalı ve isimde boşluk bulunmamalıdır. Ayrı kelimeler birleştirilmeli ve herbir alt kelime büyük harfle başlamalıdır. Örneğin; "AaaaBbbbb".

İsimlendirme ile ilgili diğer özellikler sıralanacak olursa;

- İlişkinin ismi tek olmalıdır.
- Öznitelik isimleri küçük harfle başlamalıdır.
- Öznitelik ve işlem isimlerinde karışıklıkları önlemek için açık isimler kullanılmalıdır.
- Dökümantasyon alanları geniş olarak kullanılmalıdır.
- Öznitelik isimlerinde Sınıf isimleri tekrarlanmamalıdır.
- Kısa isimler kullanılmalıdır.

UML öğelerinin isimlerinin tanımlanmasında;

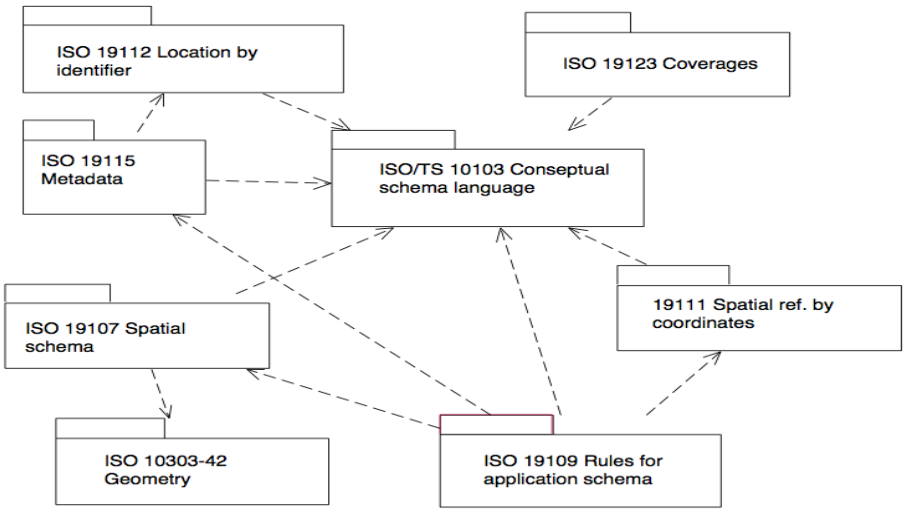
- Sınıf öznitelik işlem parametrelerinin isimleri, açık ve teknik isimler olmalıdır. Örneğin; n değil, indeks ifadesi kullanılabilir.
- Birkaç kelime varsa birleştirilmelidir. Fakat birleştirirken _ , - boşluk gibi karakterler kullanılmamalıdır. Örneğin; DrenajHavzası olmalıdır.
- Bütün öznitelik ve işlem isimleri, ilişki rolleri ve parametrelerde sadece ilk ismin baş harfi büyük olmalıdır.
- İsimlerin anlamlarını açıklamak için dökümantasyon alanları kullanılmalıdır.

İsimlerde anlaşılır ve standart kısaltmalar kullanılabilir. Örneğin; getvalue() yerine value(), Isequal yerine Equals, computeLenght() yerine length() kullanılabilir.

2.3.2.7 Paketler (Packages)

UML paketleri, sınıf ve onların ilişkilerini içeren ana taşıyıcılardır. UML'deki paket yapısı, Şekil 17'deki gibi alt paketleri ile hiyerarşik ilişkideki sınıfları ve ilişkileri tanımlar. Paket yapısı üç düzeyden oluşmaktadır;

- Düzey paketler, paketler arasındaki parçaların yapısında kullanılır.
- Düzey paketler, diğer alt paketleri içerir.
- Düzey paketler, sınıf diyagramlarını içerir.



Şekil 17 ISO/TC211 standartları arası örnek paket yapısı

2.1 Uygulama Şemasının Amacı

ISO 19109' da tanımlanmış olam uygulama şeması bir yada daha fazla uygulamanın ihtiyacı olan veri için bir konsept şemasıdır. Verinin içeriği ve yapısı uygulamadaki verinin işlemi ve manipülasyonu için operasyonların belirtimini tanımlar.

Uygulama şemasının amacı iki kısımdan oluşur:

- Veri yapısını tanımlayan bilgisayar-okunaklı veri tanımı sağlamak
- Belli uygulama anlanlarında verinin içeriğini dökümante ederek verinin ortak ve doğru biçimde anlaşılmasını gerçekleştirmek ve böylece açıkça veriden bilgi çekmeyi mümkün kılmak.

2.2 Uygulama Şeması İçin Kurallar

Bu uluslararası standartlar uygulama şemasını standardize etmez sadece tutarlı tarzda uygulama şemaları oluşturmak için kurallar tanımlar. Böylece farklı sistemler, kullanıcılar veya lokasyonlar arasında coğrafi bilginin alınmasını, işlenmesini, analizini, erişimini, sunumunu ve transferini kolaylaştırır.

Bu uluslararası standartlar veri transferi ve değişimi sırasında tedarikçiler ve coğrafi veri kullanıcıları tarafından kullanılırken

- Veri değişimi için transfer uygulama şema oluşturmak,
- Transfer edilmiş verisetinin semantiğini yorumlamak,
- İki veriseti arasındaki gerekli dönüşümlere karar vermek amaçlanmaktadır.

Bir uygulama şemasından diğer bir uygulama şemasına eşleşme, eğer iki şema çok ıraksaksa, zor hatta imkansız olabilir. Eğer sistemdeki uygulama şeması veri değişimi gereksinimleri dikkate alınarak dizayn edilmişse bu eşleşme yapmayı kolaylaştırmış olur. Bu uluslararası standartların konusu olmasa bile uygulama şeması inşa etmek için bu kurallar kullanılabilir.

Uygulama şeması oluşturmak bir işlemdir. Uygulama şemasının içeriği gerçek hayata yönelik olarak kurulmalıdır. Bu detay tipler ve özelliklerine göre modellenir.

Uygulama şeması verinin yapısını ve içeriğini tanımlar. Bu CSL(Conceptual Schema Language) içinde ifade edilir.

2.3 Veri Değişimini Destekleyen Uygulama Şeması

Bilgi sistemleri arasındaki veri değişimi iki yolla gerçekleşir

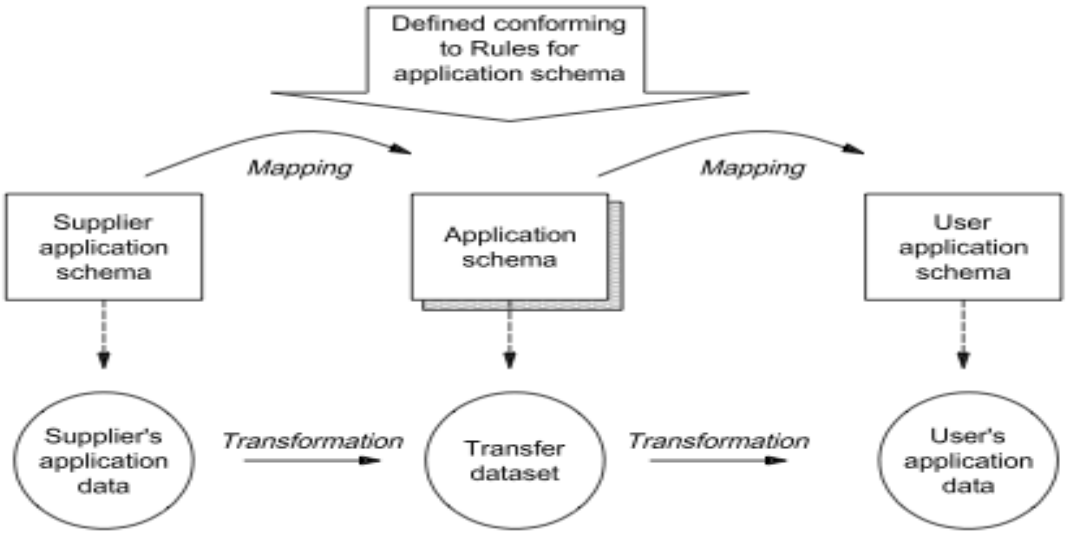
Geleneksel veri transfer modelinde veri tedarikçisi kullanıcıya transfer edilen verisetini oluşturur. Verinin yapısı ve içeriği verisetleri için uygulama şemasında tanımlanır.

Birlikte çalışabilirlik modelinde, ortak bir iletişim protokolünde kullanıcı uygulaması tedarikçi uygulaması ile iletişim kurar. Bu senaryoda kullanıcı, servis sağlayıcıdan kullanıcı uygulamasına geçecek veri için gereken servisi uyarır.

Veri transferi ile veri işlem akışı arasında temel bir ayrım vardır. Veri transferinde uygulama şemasında veriseti önceden tanımlanmıştır. Mekansal sınırlar ile ilgili kurallar da önceden tanımlanmıştır. Kullanıcı verisetinin bir kopyasını ister ve alır. (veya uzun dönem anlaşmasına bağlı olarak doğrudan alır). Veri işlem akışında istekli önce seçim kriterlerini belirtir. Veri deposundan kriterleri sağlayan veri çekilir ve istekliye sunulur.

2.3.1 Transfer Yoluyla Veri Değişimi

Şekil 18 veri tedarikçileri ve veri kullanıcıları için geleneksel veri transfer modelini gösterir. Tedarikçi tarafından sağlanan ve kullanıcı tarafından alınan verinin yapısı ve içeriği uygulama şemasında tanımlanmıştır. Veriyi transfer edebilmek için üç şartın sağlanması gereklidir.



Şekil 18 Transfer yoluyla veri değişimi

İlk olarak , kullanıcı ve tedarikçinin bu uluslararası standartlarla uyumlu ve verinin değişimi için bir uygulama şeması oluşturma üzerinde anlaşması gereklidir. Veri değişimini kolaylaştırmak için kullanıcı ve tedarikçiden uygulama şeması kullanılarak bu uygulama şemasının geliştirilmesi gereklidir.

Bu uygulama şemasına bir eşleşme tedarikçi uygulama şemasından ve bir tane eşleşme de kullanıcı uygulama şemasından yapılmalıdır.

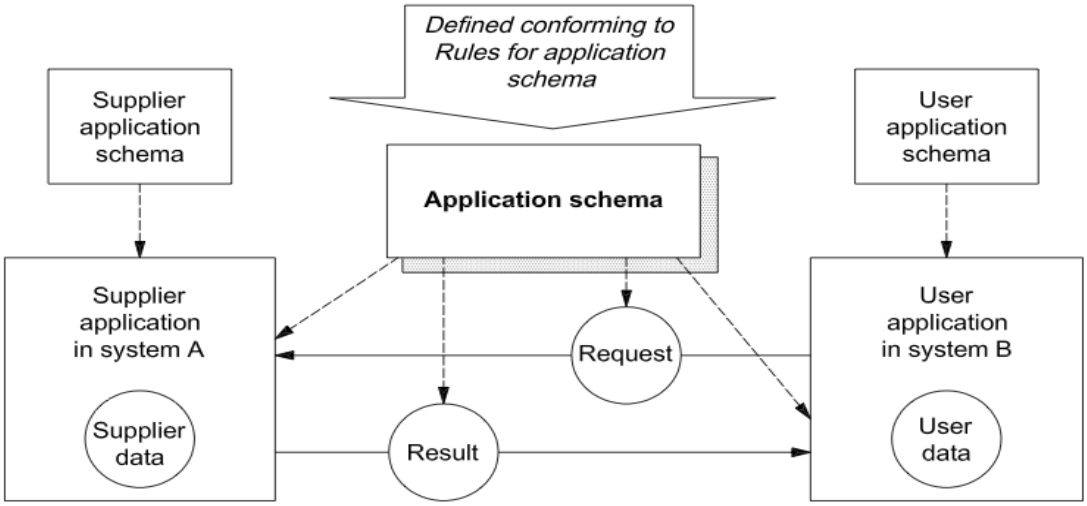
İkinci olarak, tedarikçinin tedarikçi uygulama şemasına göre tanımlanmış uygulama şemasını exchange edilmiş veri için tanımlanmış uygulama şemasına göre tanımlanmış transfer verisetine dönüştürebilmelidir.

Üçüncü olarak, kullanıcı kendi uygulama şemasına göre tanımlanmış transfer verisetini kullanıcı uygulama şemasına göre hazırlanmış uygulama verisine dönüştürebilmelidir.

2.3.2 İşlem Akışı Yoluyla Veri Değişimi

Şekil 2 birlikte çalışabilirlik modelinde anlatılan transaction yoluyla veri değişimini göstemektedir. Kullanıcı uygulaması veri için istekte bulunur. Tedarikçi uygulaması bu isteği alır ve cevap olarak bir veriseti gönderir. Hem istek hem de gönderilen veriseti ortak uygulama şemasına göre tanımlıdır.

Tedarikçi uygulaması sistem A'daki veriyi excahanged verisetindeki veriye dönüştürmek zorundadır. Verinin alımından sonra kullanıcı uygulaması exchanged veriyi sistem B'deki veriye dönüştürmek zorundadır. Transaction yoluyla veri değişimi ISO 19119da tanımlanan coğrafi servisler tarafından sağlanır.



Şekil 19 Transaction yoluyla veri değişimi

not: kırılmamış çizgiler veri akışını gösterir.kırılmış çizgiler veri değişimindeki uygulama şemasının rolünü gösterir.

2.4 Detay tanımlama prensipleri

2.4.1 Detay/Feature

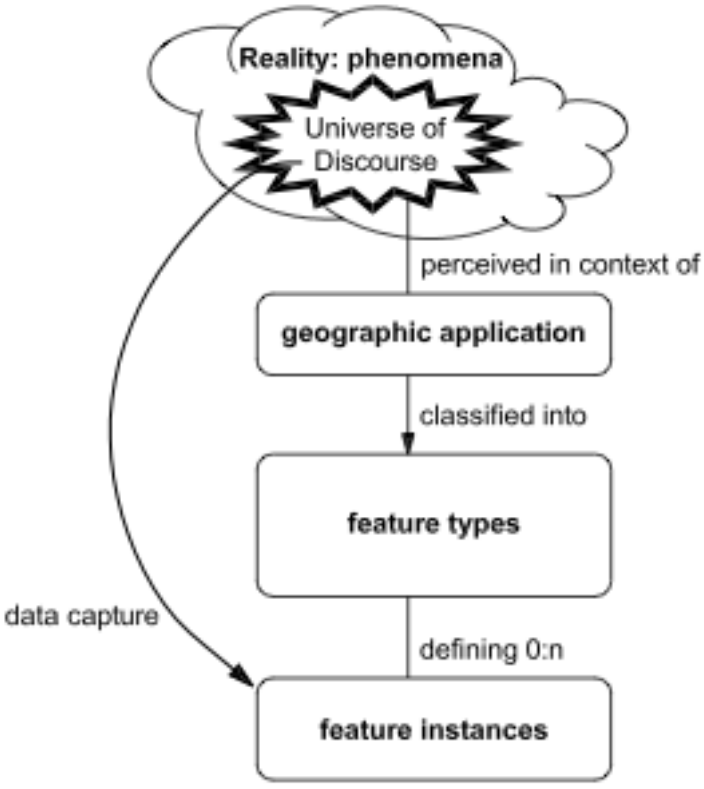
Coğrafi bilgilerin temel birimi detaydır. ISO 19110 detayların sınıflandırılmasını organize etmek ve raporlandırmak için bir standart çerçeve sağlar. Bu uluslararası standartlar, detay tanımlama prensiplerini de içeren uygulama şemaları oluşturmak için kurallar içerir. Detay dört-tabakalı mimariye göre tanımlı farklı içeriklerde kullanılır. Annex B dört tabakalı mimaride detay teriminin kullanımını anlatır.

Bu uluslararası standartlar detay tanımlamanın dört ayrı yönünü ortaya çıkarır;

- Tiplere gruplandırmak için yapılan tanımlamalar veya açıklamalar,
- Her bir tiple bağlantılı öznitelikler,
- Tipler arası ilişkiler
- Detayların davranışları.

Örneğin "Tower Bridge" Londra'da bulunan gerçek dünyaya ait köprünün soyutlamasıdır. "köprü" terimi bu terimin arkasında sınıflandırılan ve gerçek dünyaya ait olayların bir soyutlamasıdır."feature type" ve "feature instance" belli bir nesnenin varlığını tanımlayan konsepti bütün toplanmayı tanımlayan detay konseptinden ayırmak için kullanılır.

Şekil 3 coğrafi verinin tanım ve yapısının en soyut derecesini tanımlar. Gerçek dünya olaylarının detay olarak sınıflandırılması onların hedef dünyadaki hususi anlamlarına bağlıdır.



Şekil 20 Hedef dünyadan veriye doğru işlemler

2.4.2 Detaylar ve uygulama şeması

Bu uluslararası standartlar özniteliklerin tanımının uygulama şemalarına tanımlı veri yapılarına göre tanımlanmasını desteklemektedir.

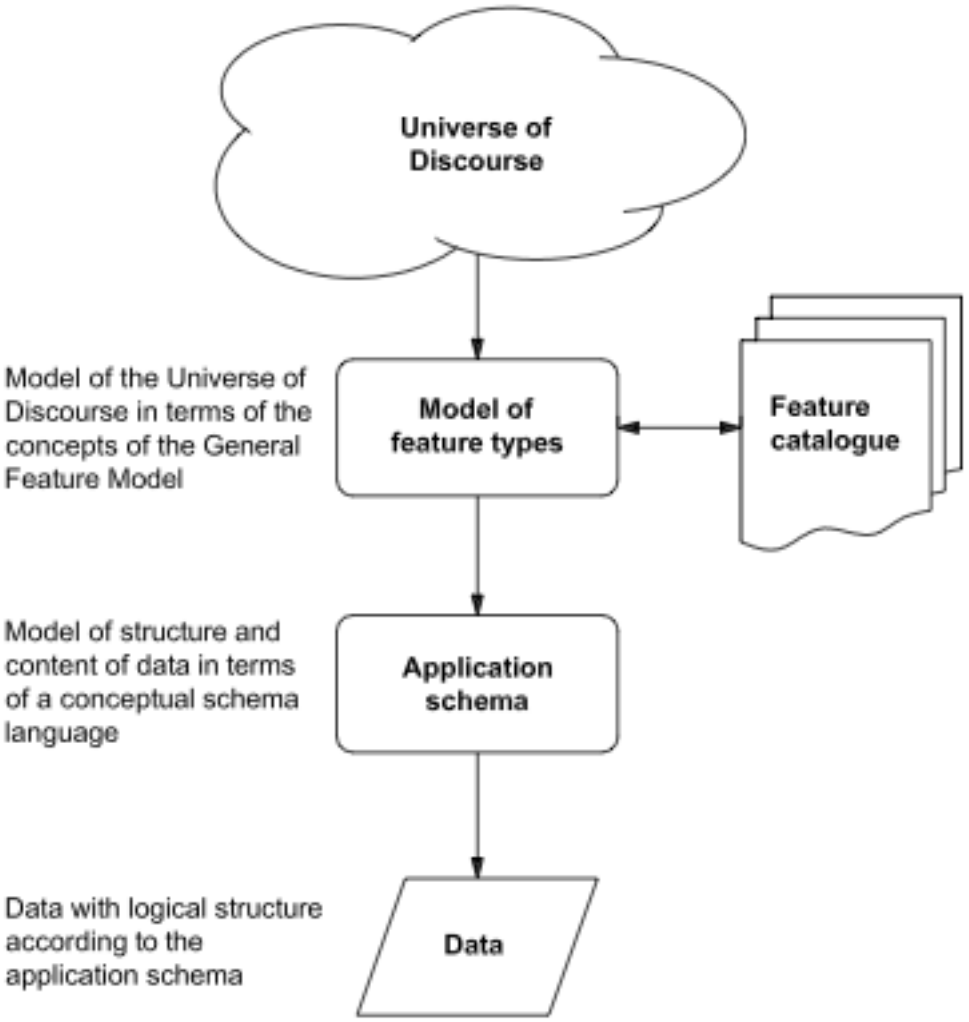
Şekil 4 hedef dünyadan coğrafi verisetine veri yapılandırma işlemini göstermektedir. Öznitelik tiplerinin ve özelliklerinin tanımı, uygulama alanı bağlamnda algılandığı üzere hedef dünyadan türetilmiş olacaktır. Detay katalog detay tiplerini dökümante eder.

Uygulama şeması verinin mantıksal yapısını tanımlar ve veriyle birlikte veya verinin üzerinde yapılacak işlemleri tanımlayabilir. Uygulama şeması fizikselden çok mantıksal organizasyonu hedefler.

Uygulama şeması geliştiricisi varolan detay katalogtaki detay tanımlarını kullanabilir. Bu veri elde etme masraflarını azaltacak, geliştiricinin varolan veriyi kullanmasına olanak sağlayacak ve uygulama şeması geliştirme işlemini kolaylaştıracaktır.

Uygulama şeması CSL (Conceptual Schema Language) içinde ifade edilmelidir. Her CSL kendi terim ve konseptlerine sahiptir. Uygulama şeması oluştururken GFM (General Feature Model) konseptleri seçilmiş olan CSL konseptleriyle eşleştirilir.

Not: Annex C GFM konseptlerinin ISO 10303-11 konseptlerine eşleşme kurallarını örnek olarak vermektedir.



Şekil 21 Realiteden coğrafi veriye

2.5 General Feature Model

Burada detay tanımlamak için kullanılan konseptleri ve bu konseptlerin birbirleriyle nasıl bir ilişki içinde olduğunu göreceğiz. General Feature Model (GFM) denen konsept modelini açıklayacağız.

Annex B GFM'in amacı ve modelini açıklamaktadır

Not: GFM'in komple özü Annex B.3 tedir.

GFM konseptleri ISO 19110 ve ISO 19117 deki detay katalog yapısında anlatılmıştır. ISO 19117 aynı zamanda bu konseptleri coğrafi bilgilerin

betimlenmesinin açıklanmasında da kullanılır. GFM konseptleri ISO 19119daki coğrafi işlem servisleri sınıflandırmasında kategori belirlemede kullanılır.7.4 te özniteliklerin farklı yönlerini 7.5 te ilişkilerin farklı yönleri anlatılmıştır. 7.6 detay davranışlarının daha detaylı anlatımını 7.7 ise kısıtlama konseptini içermektedir.

2.5.1 GFM'in Amacı

GFM gerçek dünyanın görünümünde sınıflandırma için gerekli konseptlerin bir modelidir. UML class diyagramlardaki CSL içinde ifade edilmesine ragmen herhangi bir CSL içinde de yer alabilir.UML'in kendi konsept modelleri (Metamodel) vardır

GFM ve UML her ikisi de sınıflandırmayla ilgili metamodeller olduğundan konseptler çok benzerdir. Aralarında tek büyük fark vardır: GFM özniteliklerin sınıflandırılması için bir taban olıştururken, UML metamodel herhangi bir çeşidin sınıflandırılmasında taban oluşturmaktadır.

Sınıflandırmak istediğimiz şeylere detay diyoruz ve bu detay tipleri arasındaki ilişki detay ilişki tipi ve kalıtımdır.

Detay tiplerinin detay özniteliği, detay işlemleri ve detay işlemleri rolleri şeklinde özellikleri vardır. Bütün bu konseptler GFM içindeki UML metasınıflarda ifade edildi. GFM detay tipinin metamodelidir.

Uygulama şeması CSL içinde ifade edilmelidir. Hedef dünyayı gösteren verisetinin yapısını ve içeriğini ifade eder. GFM feature sınıflandırma için gereklilikleri belirler. Bunun anlamı uygulama şeması tanımlamak için varolan bir CSL kullanmak zorundayız.

UML'de uygulama şeması yaparken GFM 'in detay sınıflandırmak için yapı tanımladığını unutmamalıyız. Fakat GFM'den UML'e eşleşme tek-yönlü eşleşme olduğundan geriye doğru eşleşme mümkün değildir. Örneğin uygulama şemasının UML sınıfları vardır. Bu sınıflardan bazıları GFM detay tiptir ve bazı veri tipleri detay öznitelik içindir. Bunları ayırmak mümkün değildir. GFM

gereken derinlikteki detay öznitelik değerlerini tanımlamaz. Buna gerek yoktur çünkü GFM sadece detay tanımlarının yapısını ve içeriğini belirtir.

2.5.2 GFM'in Ana Yapısı

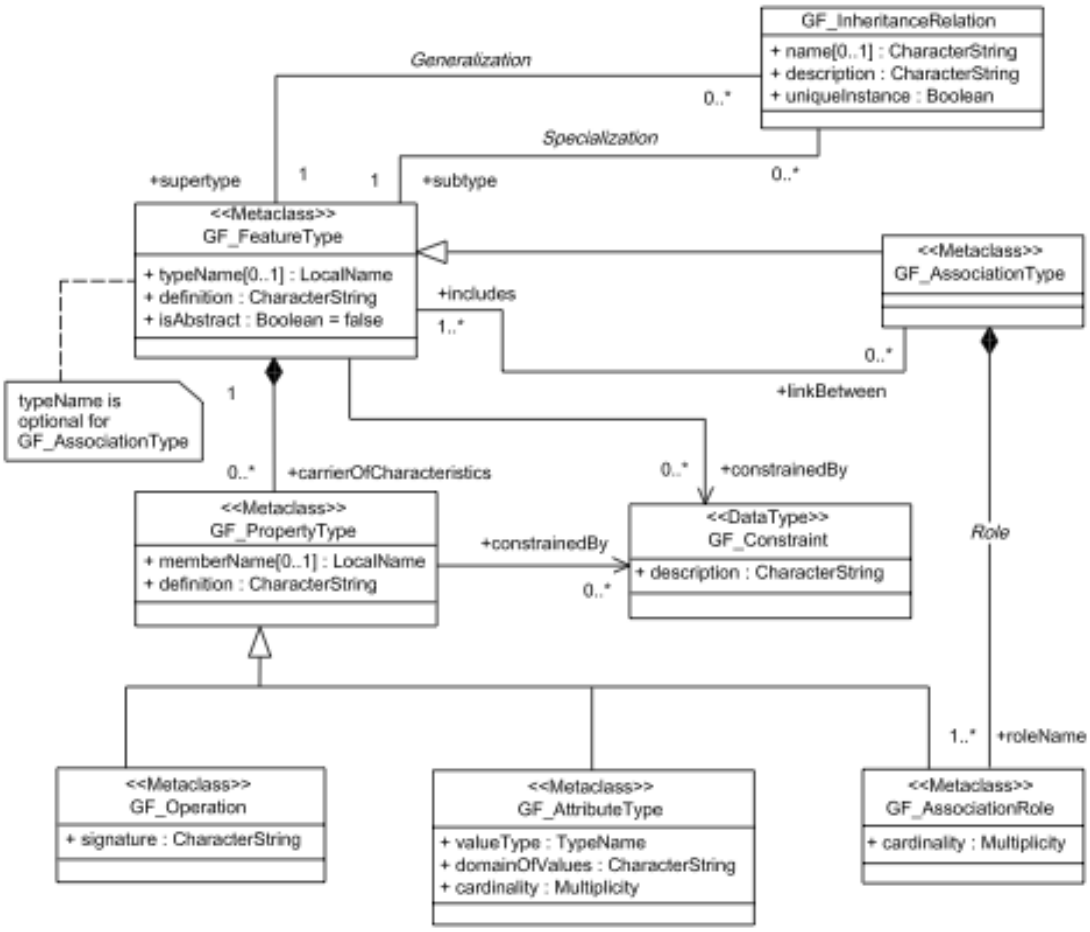
Şekil 22 detay tiplerini tanımlamak için kullanılan konseptleri gösterir, bütün modelden bir bölümdür. GFM'deki bütün konseptleri ve aralarındaki ilişkileri gösterir.

İsim ve açıklamanın dışında detay tipi için aşağıdaki özellikler de tanımlanmıştır:

- Detay öznitelikleri
- Detay tipini karakterize eden detay ilişki rolü
- Detay tipinin tanımlanmış davranışı

Bunlara ek konseptler olarak;

- Detay tipi ve kendisi arasındaki detay ilişki veya diğer detay tipleri
- Diğer detay tipleriyle generalization ve specialization ilişkileri
- Detay tipindeki kısıtlamalar



Şekil 22 General Feature Modelin bir bölümü

2.5.2.1 GF_Feature Type

Detay gerçek dünya olayının bir soyutlamasıdır. GF detay tipi ise bireysel detay tipleri gösteren metasınıftır. Belli bir detay tipi o detay tipi için bütün nesnelerin sınıfıdır. Bireysel detay tipi gösteren sınıfın nesneleri detay nesneleridir.

Not1: Detay tipini sınıflara, detay nesneleri objelere benzetebiliriz nesne tabanlı modellemedeki.

Not2: Annex B detay teriminin kullanımı için bir tablo içerir.

-typeName

Detay tipinin ismidir. Uygulama şemasında isim tek olmak zorundadır. Sadece GF_association Type'da typeName opsiyoneldir.

-LocalName

Namespace içinde yerel nesne için bir tanımlayıcıdır. ISO/TS 19103 de tanımlanmıştır. LocalName GenericName'in alttipidir

-definition

Detay tipini açıklayan tanım

-isAbstract

Boolean öznitelik. Eğer doğruysa detay tipi soyut süpertip olarak davranır.

-includes

İlişki rolü, detay ilişkisinin nesnesinin detay tipinin herhangi bir sayıda nesnesini içerebileceğini belirtir.

2.5.2.2 GF_Property Type

GF Propery Type özniteliğin karakteristiğini, davranışını veya ilişki rolünü açıklayan öznitelik tipinin özelliklerini içeren bir sınıf için metasınıftır.

GF_Property Type GF Operation, GF_ÖznitelikType ve GF_Association Role için süpertiptir.

-MemberName

Davranış, öznitelik veya rollerin isimleri. Sadece rol ismi opsiyonel.

-LocalName

Bir yerel obje için namespace içinde belirleyicidir. Bu GenericName'in hedef nesnesi olabilir, veya başka bir NameSpace(yeni bir Generic Name ile) için pointer olabilir. ISO/TS 19103te tanmlandığı üzere LocalName NameSpace tanımının bileşeni olan GenericName için bir alttiptir.

-Definition

Detay tipinin davranışının, özniteliğinin veya rolünün açıklması.

-CarrierOfCharacteristics

İlişki rolü olan CarrierOfCharacteristics; detay tipinin özniteliktipleri, işlemleri ve ilişki roller için bir belirticidir.

2.5.2.3 GF_AttributeType

GF Öznitelik tipi detay tipinin öznitelik tanımları için metasınıf'tır.

-Value type

Öznitelik değerinin veri tipidir.

Not: ISO 19103 detay özniteliklerinin değer tipi olarak kullanılan veri tiplerini tanımlar.

Örnek1: integer, character string veya GM_object

-TypeName

Yerel nesne için timespace içinde tanımlayıcıdır. Localname için alttiptir.

-Domain of Values

Değerler kümesinin açıklamaları

örnek2: pozitif, 3ten 7ye, GM_Object ve bütün alttipleri.

-Cardinality

feature tipinin tek bir nesnesine bağlı özniteliklerin nesne sayısı

-multiplicity

bir kümenin varsayımındaki izin verilebilen çokluluk aralığı

2.5.2.4 GF_Association Role

GF_AssociationType'ın bir parçası olan roller sınıfı için metasınıf'tır.

-cardinality

-multiplicity

-role

-rolename

2.5.2.5 GF_Operation

Detay tiplerinin operation bazında davranışlarını açıklayan metasınıf.

Not: GF_Operations sadece birlikte çalışabilir modellere uygulanır, veri transfer modellerine uygulanamaz.

Not2: GF_Operation instance'ları 3 çeşittir: gözlemci (observer) işlemleri, susturucu(mutator) işlemleri, inşa edici (constructor) işlemleri.

-imza (signature)kıstı

İşlemlerden dönen isim, değer ve argümanları gösteren açıklama

not3: UML'de şu şekilde belirtilir.

operation_name(input parameter 1, input_parameter2,...): output_value_type

has_height(): real gibi.

2.5.2.6 GF_Associstion Type

Öznitelik tipleri arasındaki bağlantıyı anlatan metasınıf. Detay ilişkisi öznitelik sahibi olabilir çünkü GF_Association Type GF_Feature type!ın alttipidir.

-linkBetween

2.5.2.7 GF_Inheritance Relation

genel öznitelik tipi(supertype) ile özel öznitelik tipi (subtype) arasındaki jenerik ilişki için bir sınıftır.

Örnek: feature type “bridge” hem yol feature için genel sınıf olan “transportation feature” hem de navigasyon feature için genel sınıf olan “hazards” a dahildir.

-Name

generalization/specialization ismi. Opsiyonel.

-Description

generalization/specialization açıklaması.

-Uniqueinstance

Boolean değişken.

-Generalization

-Specialization

-Supertype

-Subtype

2.5.2.8 GF_Constraint

GF constraint öznitelik tipi ve öznitelik tipi özellikleri ile bağlantılı kısıtlamalar için sınıftır.

-description

ana dilde ve resmi şekilde anlatılan kısıtlama.

-constrainedBy

kısıtlamanın GF_Feature Type veya GF_Properties üzerinde yapılmaş olmasını berlirleyen rol.

2.6 Feature Tiplerinin Öznitelikleri

Detayların öznitelikleri daha detaylı bir biçimde açıklar. Bir öznitelik tipi (GF_ Attribute Type) bir isme (memberName), bir açıklamaya (definition), bir tipe,bir domaine ve bir çokluluğa sahiptir.

2.6.1 attributeOfAtribute

bir öznitelik diğerine bağlanır ve ilk öznitelik için karakteristik özellikleri tanımlayan bağlantı.

Örnek: bir detayın pozisyonunu taşıyan öznitelik, pozisyonel tutarlılık taşıyan başka bir özniteliğe sahip olabilir.

2.6.2 GF_SpatialAttributeType

detay tiplerinin mekansal karakteristiklerini ifade eden mekânsal öznitelik.

2.6.3 GF_TemporalAttributeType

bir detay için zamansal referans karakteristiği olan geçici öznitelik ifade eder.

2.6.4 GF_QualityAttributeType

kalite bilgisi taşıyan öznitelikleri ifade eder.

2.6.5 GF_LocationAttributeType

bir detayın mekansal referansını taşıyan öznitelik

2.6.6 GF_MetadataAttributeType

metaveri bilgisi taşıyan öznitelik.

2.6.7 GF_ThematicAttributeType

yukarıdaki belirtilenlerin dışında tanımlayıcı karakteristik taşıyan öznitelik

2.7 Detay Tipleri Arasındaki İlişkiler

Detay tipleri arasındaki ilişkiyi tanımlar. Şekil 7 de ilişki şu şekilde sınıflandırılmıştır:

-Detay tiplerin generalization/specialization'ı

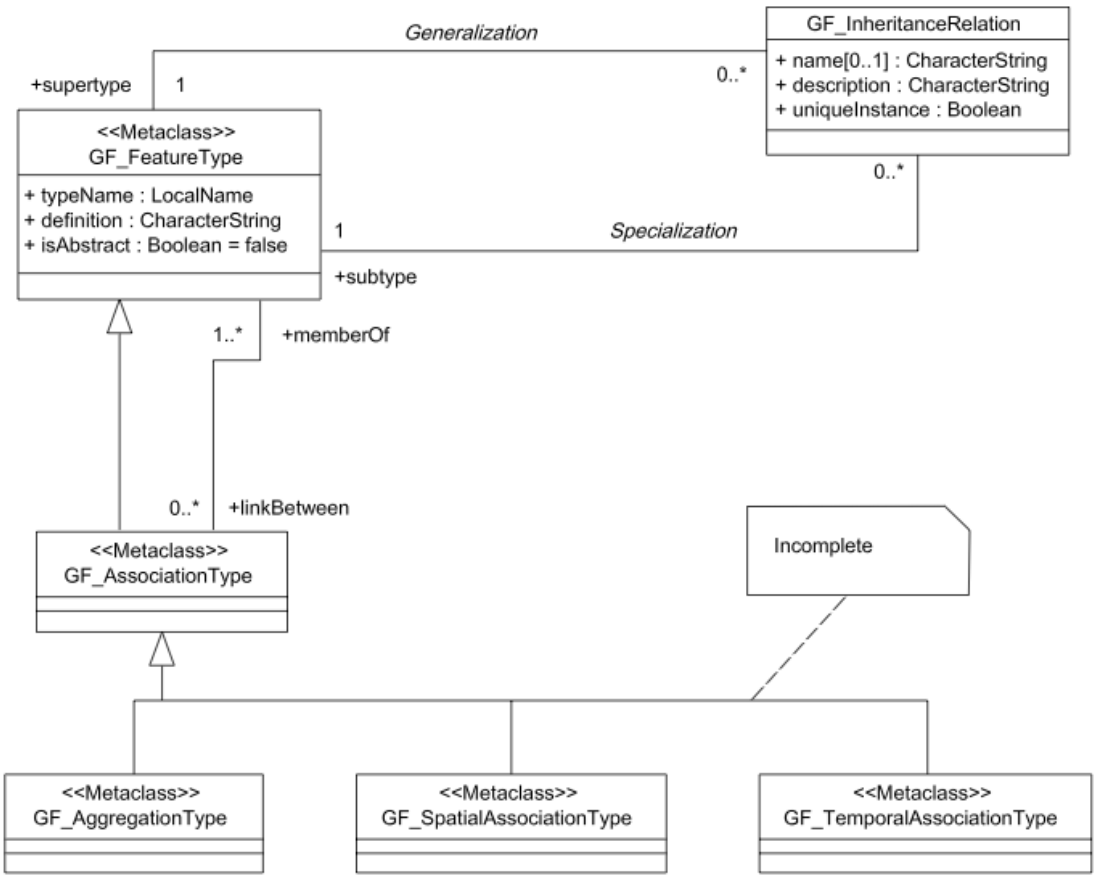
-Detaylar arasındaki bağlantı

2.7.1 GF_InheritanceRelation

Detay tiplerinden subtype ve supertype için generalization ve specialization olduklarını belirler.

2.7.2 GF_AssociationType

Detay tipleri arasındaki diğer bütün ilişki tiplerini temsil eder. Bu ilişkiler tanımladığı zaman ilişki tipi olarak ve verisette nesne olarak görünürler.



Şekil 23 Detay tipleri arasındaki ilişki

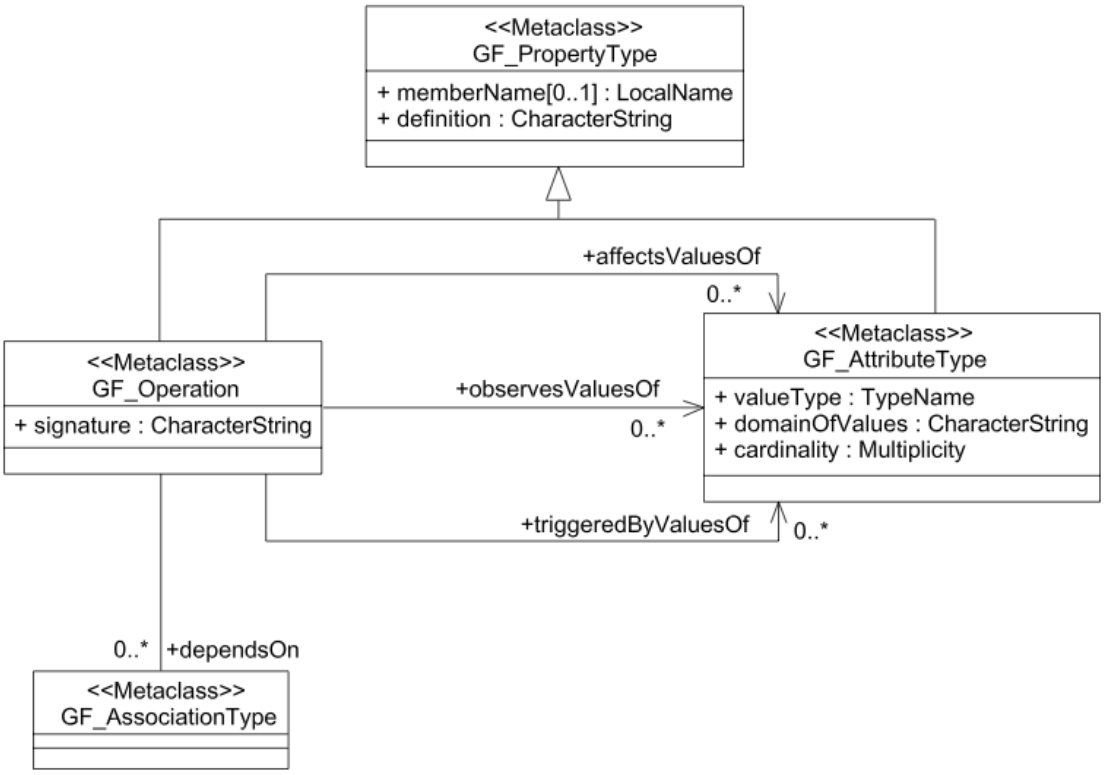
GF_association Type'ların alttiplerinden üç tanesi şöyledir:

1. -GF_Aggregation Type
2. -GF_Spatial Association Type
3. -GF_Temporal Association Type

2.8 Detay Tiplerinin Davranışları

Detay davranışı, detay tipinin nesnesi tarafından ve onun üzerinde yapılan işlemler tarafından tanımlanmıştır.GF_Operation bir detay tipin bir fonksiyon veya metod olarak davranışını gösterir.

Detay tiplerinin davranışları hakkında ISO 19110'da daha ayrıntılı olarak açıklanmıştır. Şekil 24 detay davranışları hakkında etkili olabilecek bağımlılıkları göstermektedir.



Şekil 24 Detay tiplerin davranışları

2.8.1 observesValuesOf

Bir işlem için girdi olabilecek özniteliklerin belirtildiği ilişki.

2.8.2 affectsValuesOf

Bir işlem tarafından etkilenecek öznitelikleri belirten ilişki.

2.8.3 triggeredByValuesOf

Bir işlemi tetikleyebilecek öznitelikleri belirten ilişki

2.8.4 dependsOn

Bir işlem için kullanılabilecek ilişkileri belirleyen ilişki.

2.9 Kısıtlamalar

Verinin bütünlüğünü sağlamak açısından uygulama şeması kısıtlama tanıtabilir. Kısıtlama bir uygulamadaki hatalı veri oluşmasını önler. Bunu veri kombinasyonlarının uygun olup olmadığını kontrol ederek sağlar. Bir uygulama şeması kısıtlama tanımlarını net yapmalıdır. Hem detay hem de özellikleri kısıtlamaya sahip olabilir.

Kısıtlama örnekleri aşağıdaki gibidir;

- Bir yada daha fazla detay nesnesinin (farklı tiplere ait olabilir) içindeki öznitelik değerlerinin kabul edilebilir kombinasyonlarını ayırt eder.
- Detay nesneler arasındaki ilişkilerin çokluluğunu kısıtlar
- Bir kısıtlama gerçek dünya olayının belli büyüklükte olup olmadığını gerektirebilir.
- Bir detayın davranışı detay işlemlerinde de tanımlandığı gibi kısıtlama tarafından kısıtlanabilir.

2.10 Uygulama Şeması Kuralları

Uygulama şemasının iki amacı vardır. İlk olarak belli bir uygulama alanındaki verinin içerik ve yapısının ortak ve doğru anlaşılmasını sağlar. ikinci olarak, veri yönetimi için otomatize sistemlere başvuruda bilgisayar tarafından okunabilir (computer-readable) şema sağlar.

2.11 Uygulama Modelleme Süreci

Uygulama şeması oluşturmak için adım adım işlemde iki rol gerekir. Bu adımlar kısaca şöyle açıklanabilir:

- Amaçlanan uygulama alanından (hedef dünya) gereksinimleri araştırmak
- General Feature Modelde tanımlı konseptlerle uygulamanın konsept modelini yapmak. Bu bölüm detay tipleri özelliklerini ve kısıtlamalarını da belirlemeyi içerir.
- Resmi model dilleriyle (UML ve OCL) uygulama şemasını açıklamak

- Resmi uygulama şemasını diğer standardize edilmiş şemalarla (mekansal şema, kalite şema) entegre ederek tamamen bir uygulama şeması oluşturma.

Bu işlemler iki kural seti gerektirir:

- GFM'de açıklanan konseptlerle detay tipleri nasıl haritalayacağımız
- Diğer ISO 19000 standartlarına uygun şemaları nasıl kullanacağımız

2.12 Uygulama şeması

2.12.1 Uygulama şeması için kavramsal şema

Resmi dil kullanımı modelde açık ve tutarlı gösterim sağlar. Bu da uygulamaların uygulanmasını kolaylaştırır. Bu uluslararası standartların örnek oluşturacak kısmı uygulama şemasının tanımı için resmi dil olarak UML kullanır. Bu bölümde anlatılacak kurallar UML formatına dayanır. ISO/TS 19103 stereotip ve temel veritipi kullanımını ve bunların uygulama şemaları geliştirmek için nasıl kullanıldığını gösterir.

2.12.2 Temel kurallar

Kurallar:

1) Uygulamanın veri yapısı uygulama şemasında modellenmelidir.
2) Uygulama şemasındaki veri transferi için kullanılan bütün sınıflar nesnelenebilir olmalı.

2.12.3 Uygulama şemasının tanımlaması

Kurallar:

- Her bir uygulama şemasının kimliğinde isim ve versiyon bulunmalıdır.
- UML'de bir uygulama şemasının bir PACKAGE ile tanımlı olması gerekir.

2.12.4 Uygulama şemasının dökümantasyonu

Kurallar::

1. Bir uygulama şeması dökümante edilmelidir.
2. UML'de uygulama şemasının dökümantasyonu, bu uygulama şemasını oluşturmak için kullanılan yazılım aracının dökümantasyon kolaylığından faydalanabilir.
3. Eğer bir sınıf veya diğer bir UML bileşeni detay katalogdaki bir bilgiye tekabül ediyorsa, katalog referansı da dökümante edilmelidir.
4. Uygulama şemasındaki detay tiplerin dökümantasyonu GFM'den elde edilen bir yapıyla katalogda bulunmalıdır.

2.12.5 Uygulama şeması ve standart şemanın bütünleştirilmesi

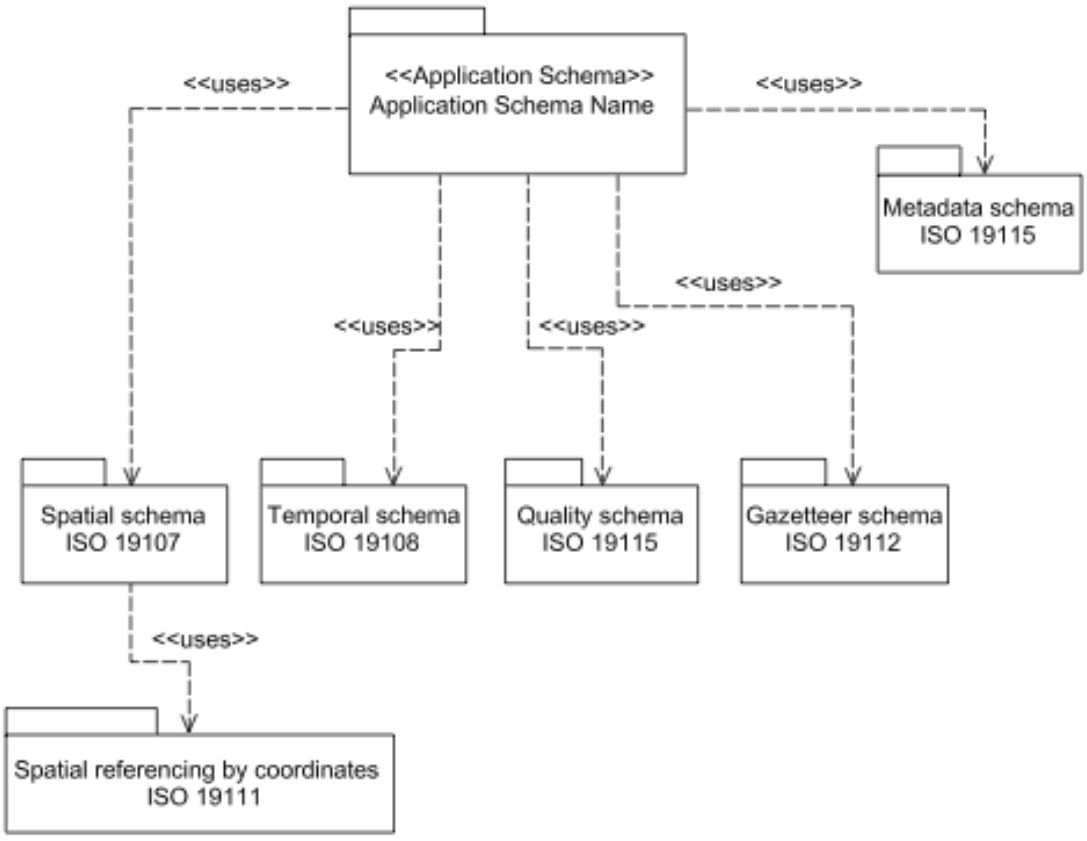
Büyük bir model oluştururken, iş genelde entegre edilebilecek bağımsız parçalara ayrılır. Bu parçalardan biri uygulama şeması olur, diğer parçalar da diğer ISO 19000 standartlarında bulunan standardize edilmiş şemalardır.

Belli bir uygulamanın veri yapısının tam tanımı, uygulamanın diğer şemalarla entegre edilmesi ve diğer şemaları doğrudans veya dolaylı olarak referans vermesinden oluşur.

Örneğin Şekil 25'de diğer ISO 19000 standartlarında bulunan şemalardan eleman kullanan bir uygulama şeması gösterilmiştir. Şekildeki bağımlılıklar uygulama şeması paketi diğer paketlerdeki tanım ve yapıyı kullanıyor "uses" anlamına gelir.

Kural:

Veri yapısının tam tanımını oluşturmak için gerekli olan uygulama şemasının diğer standart şemalarla entegresini tanımlamak için UML'de bağımlılık mekanizması kullanılır.



Şekil 25 Uygulama şeması entegrasyonu örneği

2.12.6 Yeni bir uygulama şeması oluşturmak için uygulama şemalarını kullanımı

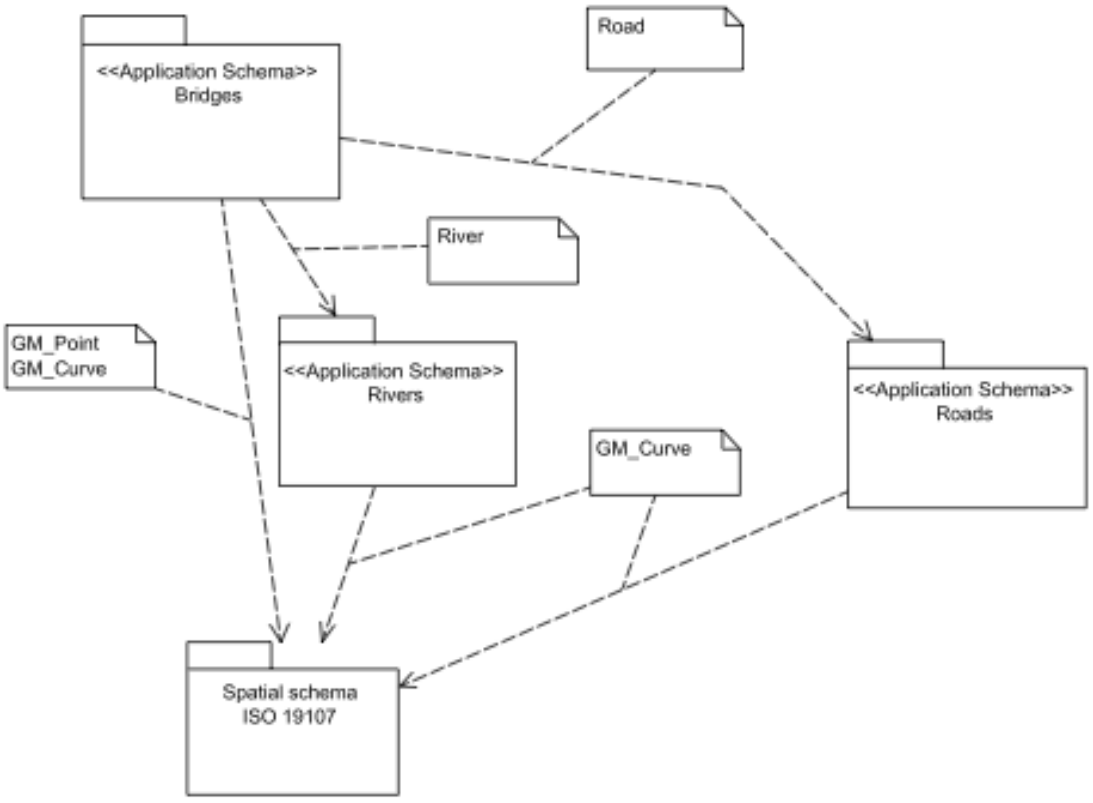
Bir uygulama şeması diğer birkaç uygulama şemasından oluşmuş olabilir. Bu şemaların her biri standardize edilmiş şemaları gösterir. Bu yapı büyük ve kompleks yapıların oluşmasını engellemeye yarar.

Örnek :Şekil 26 aşağıdaki üç UML paketi tarafından açıklanan yol, ırmak ve köprü içeren bir uygulamayı gösterir:

- -Köprüleri anlatan ana şema
- -Yol veri tipini tanımlayan şema
- -Veri tipi ırmakları tanımlayan şema

Bütün uygulama şemaları ISO 19107'deki mekansal şemadan mekansal temelleri kullanır.

2.13 UML'de Uygulama Şeması Kuralları



Şekil 26 Diğer uygulama şemalarına bağlı uygulama şeması örneği

2.13.1 Temel Kurallar

UML'de uygulama şeması için temel kurallar aşağıdaki gibidir:

Kurallar

1. GF_FeatureType: GF_FeatureType nesne sınıf olarak uygulanmalıdır. Kural 2 durum 1 hariç.
2. GF_AssocitionType: GF_AssociationType aşağıdaki 2 durumda uygulanmalıdır.

Durum1: GF_AssocistionType instance hiç bir GF_PropertyType ile bağlantılı değil. Bu durumda CLASS olarak uygulanan GF_FeatureType instance ile bağlantılı linkBetween görevi görür.

Durum2: Bir yada daha çok GF Property Type instance ile bağlantılı GF Association Type instance.ASSOCIATION CLASS olarak uygulanır.

3. GF_AggregationType: GF_Aggregation Type nesnesi AGGREGATION(içi boş elmas) yada COMPOSITION(içi dolu elmas) olarak kullanılmalıdır.
4. GF_Attribute Type: GF_Attribute Type eğer özniteliğin özniteliği değilse ÖZNİTELİK olarak kullanılmalıdır.
5. attributeOfAttribute: GF_Attribute Type'ın attributeOfAttribute bağlantısındaki nesnesi sınıf olarak kullanılmalıdır.
6. GF_Operation:GF_Operation nesnesi karakterize ettiği detay tipinde OPERATION olarak kullanılmalıdır.
7. GF_AssociationRole: GF_AssociationRole bağlantıların uygun olan uçlarında rol ismi olarak kullanılmalıdır.
8. GF_InheritanceRelation: GF_InheritanceRelation UML GENERALIATION ilişkisi olarak gösterilmelidir.
9. GF_Constraint: kısıtlamalar kısıtlanmış olan SINIF, İŞLEM veya İLİŞKİLERE sade bir dilde eklenmelidir.

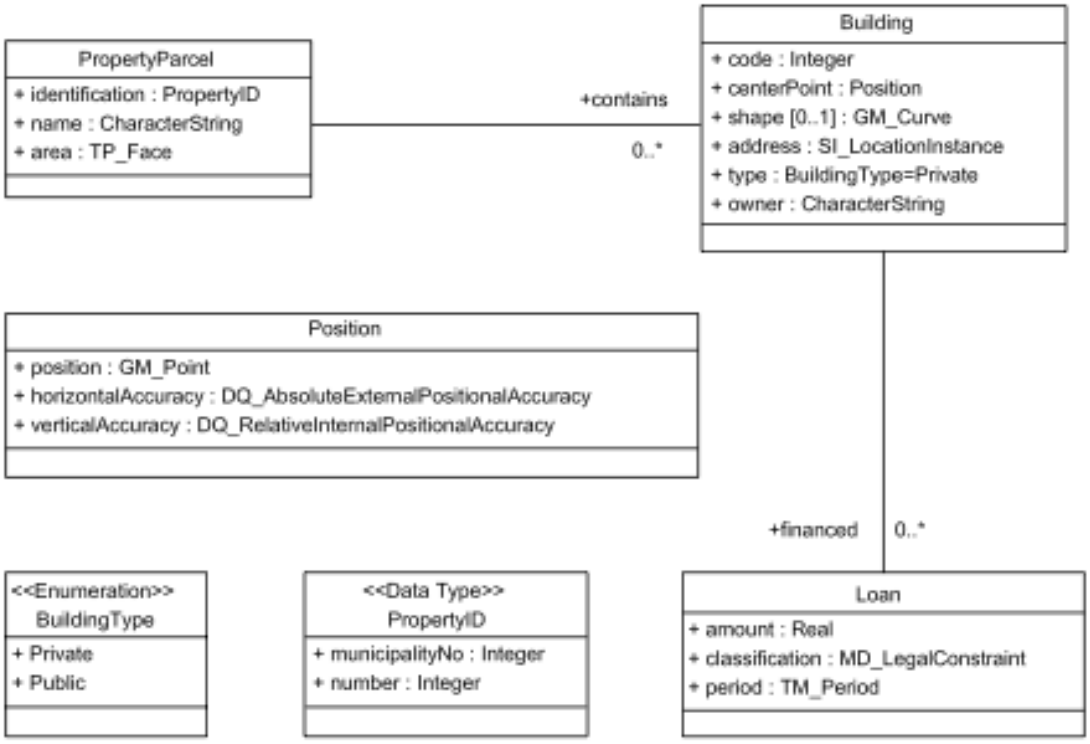
2.13.2 Uygulama Şeması Örnekleri

Şekil 28' da Şekil 27'teki gerçek dünya konseptleri tabanlı UML'de hazırlanmış bir uygulama şeması örneği gösterilmiştir.

Feature types	Attributes	Kind of subtype of GF_Attribute
Property parcel	Identification Name Area	Thematic attribute type Thematic attribute type Spatial attribute type
Building	Code Centre point Shape Address Type Horizontal accuracy Vertical accuracy Owner	Thematic attribute type Spatial attribute type Spatial attribute type Location attribute type Thematic attribute type Quality attribute type Quality attribute type Thematic attribute type
Loan	Amount Period Classification	Thematic attribute type Temporal attribute type Metadata attribute type

Şekil 27 General Feature Model ile gerçek dünya konsepti örneği

Yatay ve düşey doğruluk öznitelikleri orta nokta özniteliğinin kalitesini açıklar. İki tane bağlantı vardır: *Property parcel* sıfır, bir veya çok *Buildings* içerir. Bir *Building* sıfır, bir veya çok *Loans* tarafından finance edilmiştir.



Şekil 28 Detay tiplerin UML uygulama örneği

KAYNAKLAR

Aydinoglu A.C., Yomralıoğlu T., 2010. "Harmonized Geo-Information Model for Urban Governance", ICE-Municipal Engineer, Vol. 163, No. 2, 06/2010, s. 65-76.

Aydınoğlu, A.Ç., 2009. "Türkiye için Coğrafi Veri Değişim Modelinin Geliştirilmesi", KTÜ Fen Bilimleri Enstitüsü, Doktora Tezi, Trabzon.

Aydınoğlu A.Ç., Yomralıoğlu T., 2011. "Coğrafi Verilerin Birlikte Çalışabilirliğine Yönelik Veri Değişim Modelinin Geliştirilmesi", 2011, HKMO 13. Türkiye Harita Bilimsel ve Teknik Kurultayı, Ankara, 18.04.2011 - 22.04.2011.

EuoGeographics, 2008. Eurogeographics Web Sitesi, www.eurogeographics.org, 07.11.2008.

INSPIRE DT, 2007. D2.6: Methodology for the development of data specifications, INSPIRE Drafting Team Data Specifications D2.6, Ispra.

INSPIRE DT, 2008. Questionnaire on Unique Identifiers- Report, INSPIRE Data Specifications, Ispra.

ISO/IEC, 1996. ISO/IEC 1157 Information technology -- Open Systems Interconnection -- Remote Procedure Call (RPC).

ISO/TC211, 2002. "ISO 19113 Geographic Information/Geomatics- Quality principles".

ISO/TC211, 2003. "ISO 19123 Geographic information — Schema for coverage geometry and functions".

ISO/TC211, 2006. "ISO 19109 Geographic Information- Rules for application schemas".

ISO/TC211, 2005b. "ISO 19103 Geographic Information — Conceptual schema language".

ISO/TC211, 2005c. "ISO 19108 Geographic information — Temporal schema ".

ISO/TC211, 2005d. "ISO 19107 Geographic information — Spatial schema".

Nebert, DD, 2004. Nebert, D.D., 2004. Developing Spatial Data Infrastructures:The SDI Cookbook Version 2, GSDI.

NEN, 2005. NEN 3610: Basic model for geo-information - Terms, definitions, relations and general rules for the interchange of information of spatial objects related to the earth's surface, NEN Standart Committee Geographic Information, ICS 01.040.93; 91.020.

Neudeck, S., 2001. Stefan: Zur Gestaltung topographischer Karten für die Bildschirmvisualisierung. Universität der Bundeswehr. Studiengang Geodäsie und Geoinformation. München.

Robinson, A., Morrison, J.L., Muehrcke, P.C., Kimerling, A.J. ve Guptill, S.C., 1995. Elements of Cartography. John Wiley & Sons, Inc.

Stoter, J., 2006. State-of-the-art of generalization within NMA's, INSPIRE Workshop on Multiple-Representation and Data Consistency, ITC, The Netherlands.

T.C. Resmi Gazete, Harita ve Harita Bilgilerini Temin ve Kullanma Yönetmeliği (22037), 31.08.1994.

T.C. Resmi Gazete, 2005. BÖHHBÜY-Büyük Ölçekli Harita ve Harita Bilgileri Üretim Yönetmeliği. (25876), 15.07.2005.

USGS, 2005. Project Bluebook: NSDI Stewardship Guidance, 1-3, July, USA.

JRC, 2012. "A Conceptual Model for Developing Interoperability Specifications in Spatial Data Infrastructures", EU JRC Reference Report