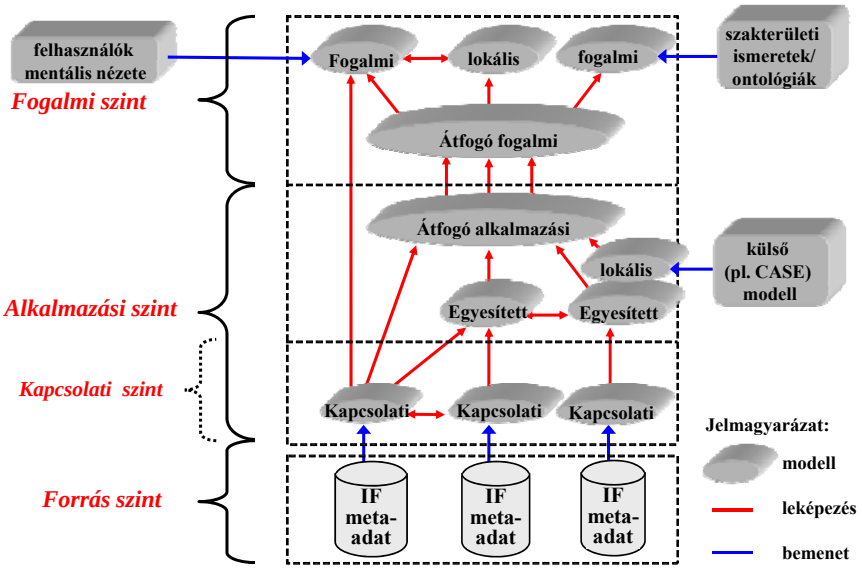
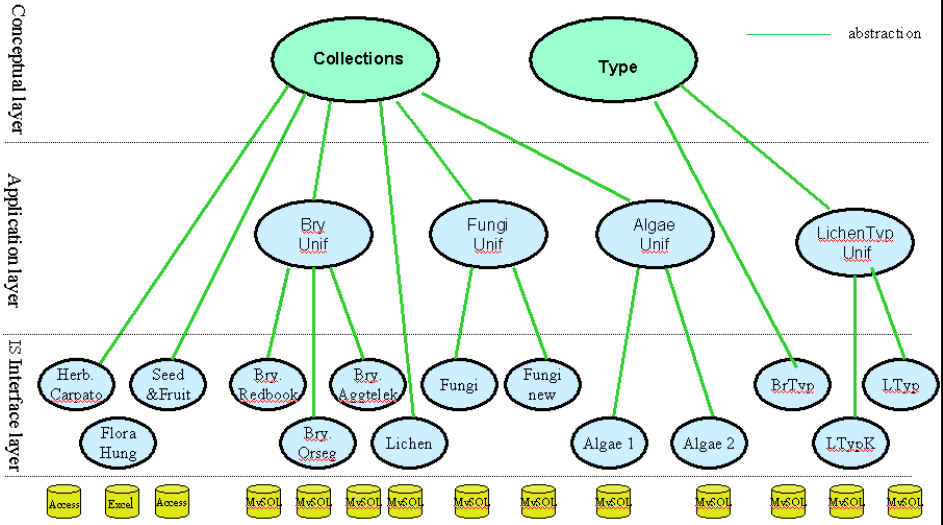
Információforrások

- Bármi, ami objektumorientált modellel leírható
- Példák
  - relációs adatbázisok
  - objektumorientált adatbázisok
  - XML fájlok
  - Web-szolgáltatások
  - RDF fájlok

Modelltárház szerkezete



A BOTANIKA alkalmazás modelliei



Modellreprezentációs nyelvek. A CORBA IDL

```
• module Server {
    interface Face : BaseFace {
      readonly attribute TypeCode type;

      string sayHello(in memberS m);
    };

    struct Member {
      string name;
      TypeCode type; };

    const long MaxLen=64;

    enum Boolean { true, false };

    typedef sequence <Member> memberS;

    typedef string ScopedName;
  };
```

Modellreprezentációs nyelvek. A CORBA IDL

- module: szoftver csomagok
- interface: osztályok attribútumokkal és eljárásokkal
- struct: „könnyű”, tranziens osztályok, csak adatelemekkel
- enum: felsorolások
- typedef: típusdefiníciók
- const: konstansok
- sequence: vektorok

SILan – a SILK tudásreprezentációs nyelv

- A modelltárház szöveges megjelenési formája
  - modellek (korlátokkal)
  - leképezések
  - lekérdezések
- Meglévő (szabványos) nyelvek újrahasznosítása
  - CORBA IDL a modellekhez (az UML nem definiál szöveges alakot)
  - OCL a korlátokhoz
  - SQL és OQL a lekérdezésekhez (OCL-lel a *where* részben)
- Példa

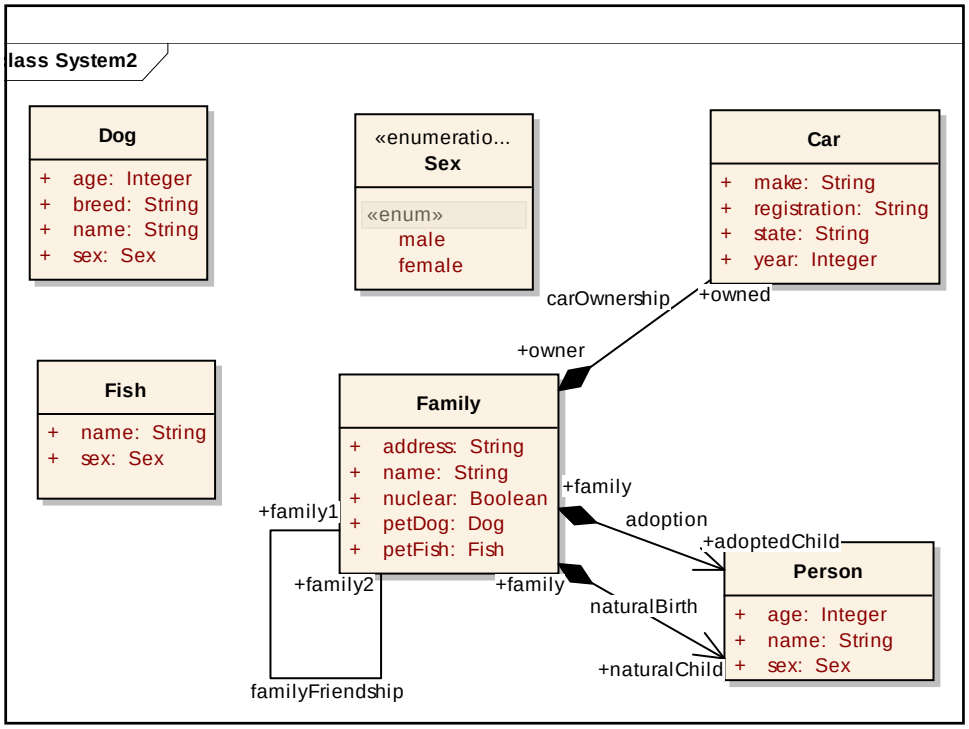
```
model Finance {
  class Employee {
    attribute String firstname, lastname;
    attribute Real salary, tax;
    constraint tax = 0.25*salary and tax >= 10000;
    primary key (firstname, lastname);
  };
};
```

CORBA IDL

OMG OCL

Human Usable Textual Notation (HUTN)

- Példány és modell egyöntetű kezelése: példányszerűen, a modell a metamodell példánya
- Konfigurálható – konkrét nyelvtan nem rögzített
  - Boolean vagy Enum értékek használata az osztályfejben, mintegy jelző szerűen
  - Attribútumok használata paraméter-szerűen
- Tartalmazott objektum ábrázolható beágyazva (azonosító név használata elhagyható)
- Osztálypéldányok azonosítása: valamely attribútumukkal – mely hatáskörön belül egyértelmű?
  - osztály+alosztályok? tartalmazó objektum?
  - Egyszerűsített azonosító nevek használata
- Alapértelmezett értékek: mely attribútumok elhagyhatók, mert default értékük van…
- Előre definiált konfigurációk…



```
FamilyPackage id-001 {
  Family “The McDonalds” {
    address: “7 Main Street”
    migrants
    familyFriends:
                “The Smiths”
    petFish: female Fish “Wanda”;
    petDog: “Spike”
    CarOwnership: “755-BDL” {
      state: QLD
      make: “Mitsubishi Magna”
      year: 1992 }
  }
  nuclear Family “The Smiths” {
  address: “5 Main Street”
  naturalChild: female Person
    “Joan Smith” {age: 20}
  naturalChild: male Person
    “Harry Smith” {age: 17 }
  adoptedChild: male Person
    “Dylan Smith” {age: 12 }
  familyFriends: “The McDonalds”
  }
  male Person “Namdou Diaye”
    {age: 6 }
  male Person
    “Sharif Mbangwa” {age:3}
  male Person
    “Miguel Aranjuez” {age:2}
  male Dog “Spike” {
    age: 2
    breed: “Irish Wolfhound”
  }
```

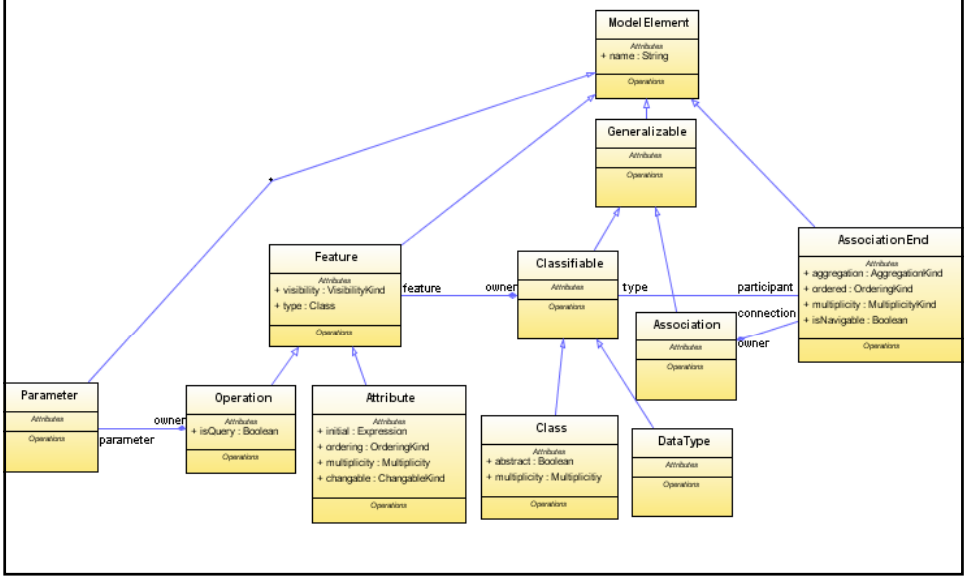
Metanemterminális-készlet

- LISTOF(NT, Nyitó, Elválasztó, Záró)
- IFTHEN(Boolean,THEN_NT,ELSE_NT)
- IFTHELSE(Boolean, THEN_NT, ELSE_NT)

Metamodellvezérelt mentés-betöltés

- Kétszintű (pongyola) nyelvtanok: CFG+metakonstrukciók, pl…
- ```
  class package {…
    attribute String
      syntax:[comment,„package”,
        name, LISTOF(ownedElement, „{”,„;”, „}”)];… };
  ```
- ```
  class class {…
    attribute String
      syntax:[comment, „class”, name,
        LISTOF(feature, „{”,„;”, „}”)];… };
  ```
- ```
  class attribute {…
    attribute String
      syntax:[„attribute”, name, multiplicity, type];… };
  ```
- Metamodell elemek←→nemterminálisok (nyelvtanleírás pl. név-érték párokkal)
- Mentés/betöltés: metamodell elemek meginterpretálása…

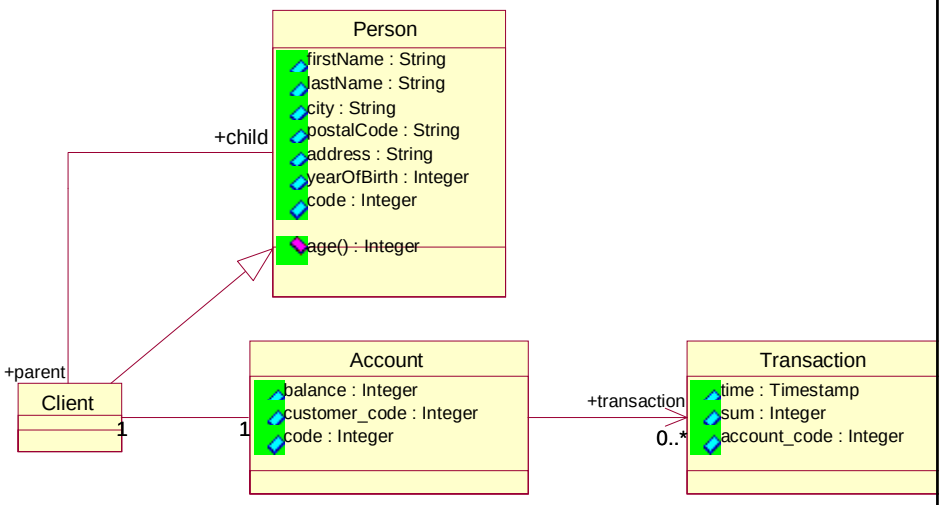
Egy UML rész-metamodell


  ```

Lekérdező résznyelv: (OQL)

- Szabványos nyelv (Object Data Management Group)
- Csak objektum-orientált fogalmakból építkezik
- SQL kiterjesztés
- (egyszerűsített) OQL (EBNF) nyelvtan:
- <lekérdezés>:=„select” [<mit>]* „from” [<honnan>]* „where” <feltétel>.
- <mit>:=<kifejezés>{, <kifejezés>}*
- <honnan>:=<típus>{, <típus>}*
- <feltétel>:=<kifejezés> logikai értékkel
- <kifejezés>:=<literális>|<változó>|<konstans>|<típus>
- <kifejezés>:=<kifejezés>.<feature>
- <feature>:=<tulajdonság>|<kapcsolatvég>
- <feature>:=<módszer>({<kifejezés>{,<kifejezés>}})

OQL példák. UML Modell.



OQL Példák

- 100 évnél öregebb ügyfelek kiválasztása:
- `select firstName, lastName, postalCode, city from Client where age()>= 100;`
- A háromnál több gyerekes ügyfelek kiválasztása
- `select firstName, lastName, child.firstName from Client where child->size()>3;`
- Gyanúsan nagyösszegű tranzakció a gyerekek számláján
- ```
  select firstName, lastName,
     (select child.firstName, child.trans.date
      from Client where child.age()<18 and
      child.trans.sum>2000000)
  from Client;
  ```

Model Verifier

- Feladat
  - modellelemek egy halmazának konzisztenciáját ellenőrizni
  - az OCL ekvivalens az elsőrendű logikával → teljesség nem várható el
- Megvalósítás
  - mélység és időkorlátos végrehajtás
  - moduláris
    - több korlát-megoldó (CLP(R), {log}, CHR kiegészítések)
    - ütemező koordinálja a megoldókat
  - a válasz ellentmondásos korlátok egy halmaza
  ```

Példa az összehasonlításra és ellenőrzésre I.

•A modellek

```
model Finance {
  class Employee {
    attribute String firstname;
    attribute String lastname;
    attribute Real salary, tax;
    constraint tax = 0.25*salary;
    constraint tax >= 10000;
    primary key (firstname,
                 lastname);
  };
};
```

```
model Production {
  class Worker {
    attribute String name;

    attribute Real salary;
    attribute String skills;
    constraint salary < 400;
    primary key name;

  };
};
```

•Automatikusan generált leképezés (nem lefordítható a **'function?'** miatt)

```
map bundle between Finance and Production {
  correspondence (e: Finance::Employee, w: Production::Worker) {
    constraint w.name = 'function?'(e.lastname, e.firstname)
       implies w.salary = e.salary;
    tagged value status = "generated";
  };
};
```

Példa az összehasonlításra és ellenőrzésre II.

•Az első javított leképezés

```
map bundle between Finance and Production {
  correspondence (e: Finance::Employee, w: Production::Worker) {
    constraint w.name = e.firstname.concat(e.lastname)
       implies w.salary = e.salary;
    tagged value status = "edited";
  };
};
```

•Jelentés

- •egy **Employee** megfelel egy **Worker**-nek, ha az **Employee** teljes neve megegyezik a **Worker** nevével
- •ha egy **Employee** megfelel egy **Worker**-nek, akkor a fizetésük egyenlő

•A Model Verifier által észlelt inkonzisztencia: az adó túl magas, a fizetés túl alacsony

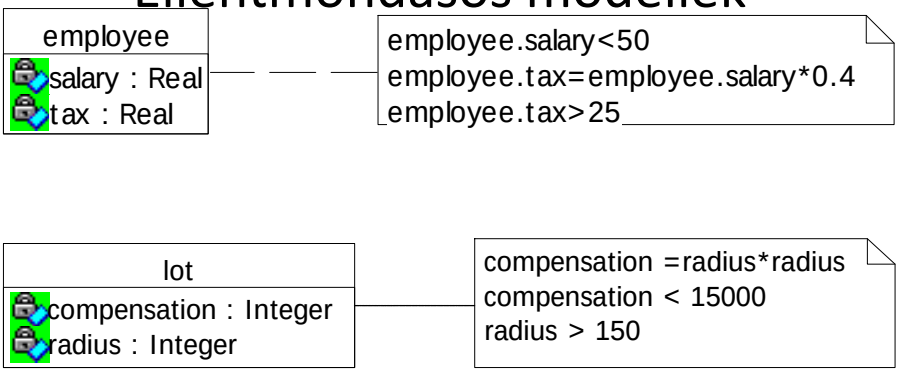
```
e.tax = 0.25*e.salary, e.tax >= 10000,
w.salary = e.salary, w.salary < 400
```

Példa az összehasonlításra és ellenőrzésre III.

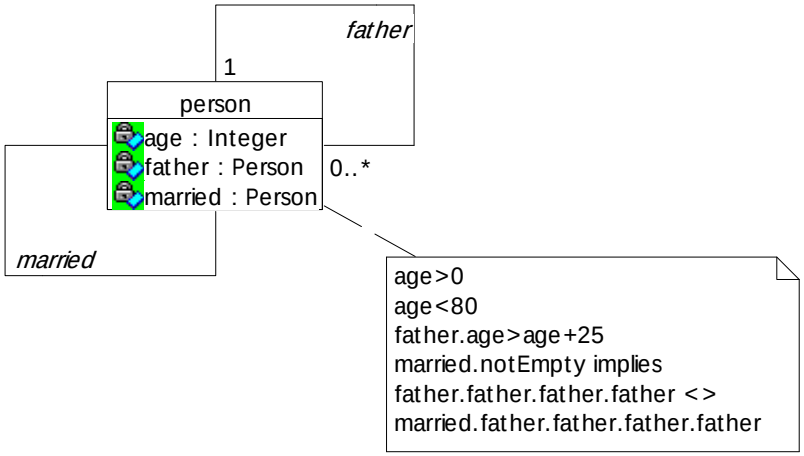
•Egy lehetséges javítás

```
map bundle between Finance and Production {
  correspondence (e: Finance::Employee, w: Production::Worker) {
    constraint w.name = e.firstname.concat(e.lastname)
       implies w.salary*1000 = e.salary;
    tagged value status = ”edited”;
  };
};
```

Ellentmondásos modellek



Ellentmondásos modellek



Ellentmondás öröklődésen keresztül