

# ATLAS Transformation Language

## ATL

- OCL-re épülő nyelv, transzformációk leírására
- A transzformációt végrehajtani képes ATL Virtuális Gép specifikációja és implementációja
- Leképezés az ATL nyelvről erre a „gépi kód”-ra.

## További eszközök

- ADT: ATL Development Tools
  - Eclipse-re épülő ATL program szerkesztő, szintaxis ellenőrzéssel, automatikus fordítással
  - Debug támogatás
- KM3: Kernel Meta-MetaModell
  - Egyszerű szöveges, kicsit java-ra hasonlító nyelv metamodellek leírására
  - KM3-ből Ecore, MOF metamodell létrehozása

## KM3 támogatás

- ECore, MOF metamodell generálása
- KM3 metamodell ellenőrzése:  
KM32Problem.atl

## KM3 Metamodellek

```
package Book {
  class Book {
    attribute title : String;
    reference chapters[*] ordered container : Chapter oppositeOf
  }
  book;

  class Chapter {
    attribute title : String;
    attribute nbPages : Integer;
    attribute author : String;
    reference book : Book oppositeOf chapters;
  }
}

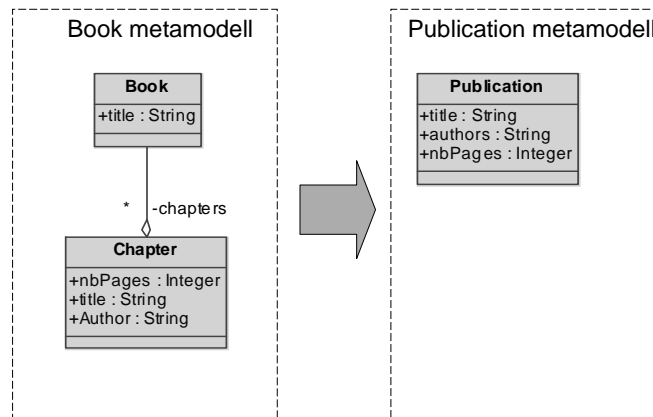
package PrimitiveTypes {
  datatype Integer;
  datatype String;
  datatype Boolean;
}

enumeration Color {
  literal green;
  literal blue;
  literal orange;
}
```

## ATL fájlok

- **ATL module:** egy vagy több bemeneti és egy kimeneti metamodell közötti transzformáció megadása
- **ATL query:** egy bemeneti modell transzformációja valamilyen szöveggé, pl. kódgenerálás
- **ATL library:** hasznos helper függvények összefogása egy újrahasznosítható könyvtárba, pl. strings.atl

## Transzformációk



## Transzformációk

- ATL modul: 'module' kulcsszó + modulnév  
`module Book2Publication;`
- Be/kimeneti modellek, metamodellek megadása:  
`create OUT : Publication from IN : Book;`
- Modulok importja:  
`uses strings;`

## Transzformációk

- Helper függvények

```
helper context Book!Book def : getAuthors() : String =
  self.chapters->collect(e | e.author)-> asSet()->
    iterate(authorName; acc : String = '' |
      acc +
        if acc = ''
          then authorName
          else ' and ' + authorName
        endif
    );
```

## Transzformációk

- Szabályok

```
rule Book2Publication {
  from
    b : Book!Book (
      b.getNbPages() > 2
    )
  to
    out : Publication!Publication (
      title <- b.title,
      authors <- b.getAuthors(),
      nbPages <- b.getNbPages()
    )
}
```

## Transzformációk

- Egy lépésben több célmodell elem is létrehozható:

```
rule Class2Table {
  from
    c : Class!Class
  to
    table : Relational!Table (
      name <- c.name,
      key <- Set {new_key}
    ),
    new_key : Relational!Column (
      name <- 'objectId'
    )
}
```

## Transzformációk

- A létrehozott elemek száma lehet dinamikus:

```
rule Root2List {
  from
    f : Folder!File (
      f.folder.oclIsUndefined()
    )
  using {
    allFiles : Sequence(Folder!File) = Folder!File.allInstances()->
      select(e | e.oclIsTypeOf(Folder!File))->asSequence();
  }
  to
    out : List!List (
      name <- f.name
    ),
    fi : distinct List!File foreach(singleFile in allFiles) (
      name <- singleFile.name,
      list <- out
    )
}
```

## Lekérdezések

- A lekérdezéseknek nincs kötött kimeneti modelljük, ezért pl. kódgenerálásra használhatóak:

```
query XQuery2Code = XQuery!XQueryProgram.allInstances()
->collect(e | e.toString().writeTo('C:/test.xquery'));
helper context XQuery!XQueryProgram def: toString() : String
= self.expressions->iterate
(e; acc : String = '' | acc + e.toString() + '\n' );

helper context XQuery!ReturnXPath def: toString() : String =
{' + self.value + '};
helper context XQuery!BooleanExp def: toString() : String =
self.value;
```

## Könyvtárak

- Globális függvények, változók

```
library GeometryLib;
helper def: PIDiv180 : Real = 180.toRadians() / 180;

helper def : addVectors (
a : TupleType(x : Real, y : Real, z : Real),
b : TupleType(x : Real, y : Real, z : Real)):
TupleType(x : Real, y : Real, z : Real) =

Tuple {
x = a.x + b.x,
y = a.y + b.y,
z = a.z + b.z
};
```

## Könyvtárak

- Importálás után a `thisModule` előtag segítségével hivatkozhatunk a könyvtár elemeire:

```
thisModule.PIDiv180
```

```
thisModule.addVectors (  
  Tuple {x=10, y=10, z=10},  
  Tuple {x=10, y=10, z=10}  
)
```