Code generation, reverse engineering

19/11/2018

Goal :

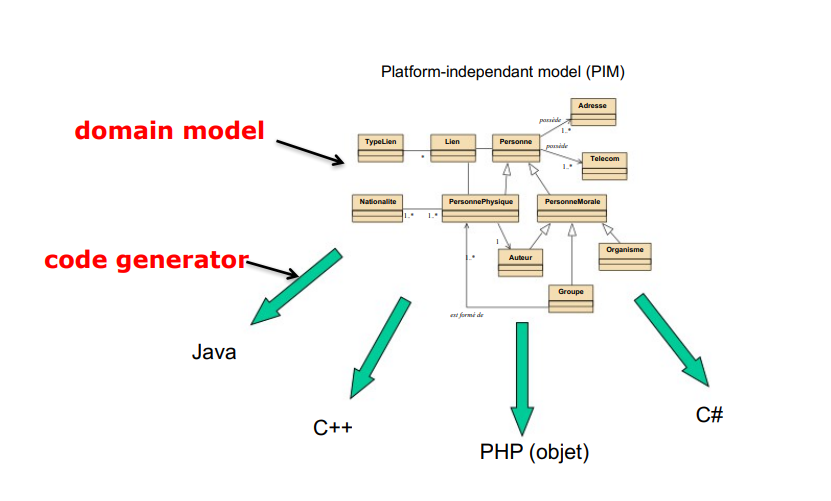
* increase productivity : you can generate lots of things like databases, configuration files….
* used in big projects
* quality gain

Code generation differs with the tools you use.

MDE vision : one model, many implementations

The software developer is replaced by :

* domain model creator
* code generator developer



code generator :

* input : UML Model
* output : code

types in UML ⇔ Java

|  |  |
| --- | --- |
| UML Types | Java |
| String  integer  real  boolean  char | String  int  double, float…  boolean  char |

For each class, you have to find the equivalent type

|  |  |  |
| --- | --- | --- |
| (In | p : | Personne) |
|  | name | type |

3 types :

* in : the value of this data will be provided
* out : return
* inout :

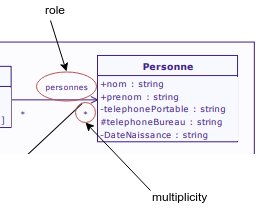
out isn’t supported by java

**A -> B**

* A source
* in the code, you have a new attribute in the source class (B generates an attribute in A).

In the code, the type depends on the multiplicity, and the name is the role.

eg :



this relationship becomes :

*public List personnes = new ArrayList();*

In Repertoire class (diapo 13)

inheritance => extends

Using the different UML models enables you to check if your the way you thought your program is correct.

Activity diagrams :

* slower
* no generators, only prototypes
* but it helps you to see if there is any errors or misconceptions

code generation & operations : instead of using activity diagrams, you can add code notes to your generator. The problem is that your model isn’t flexible anymore, and will be adapted only for one language.

static : the same shared value between objects.

ex : nb of students

**reverse engineering :**

input : code

output : model

Model =/= diagram

A model comprises all the elements to make one or several diagrams (usecase, class diagram, etc etc)

Steps : source code => model => create diagrams

**Round trip engineering**

From model to code and back again

Synchronization

It is different than Generation + Reverse

**Documentation generation**

rt.jar => contains all of the compiled class files for the base Java Runtime environment.