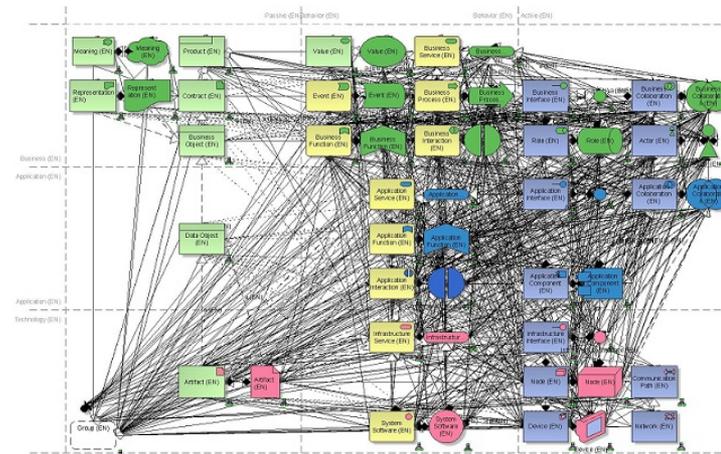


Demystifying Metamodels



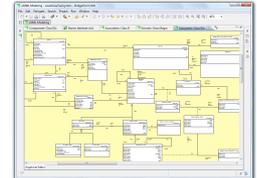
Just a sample metamodel from the web...

Metamodel – What is it ?



- What is a metamodel?
 - Who knows about metamodels? (typically only a few)
 - Since this is a slide show mainly about metamodeling and transformations between metamodels, there is a need for a short intro on metamodels, in order for the rest of the slides to make at least some sense to beginners.
- Metamodel definition:
 - Wikipedia:
A meta-model typically defines the language and processes from which to form a model.
 - Steve Mellor:
A *metamodel* is a model of a language expressed using a modelling language.
(= it's a model of the modeling language)
 - For example, the model of Executable UML is expressed using Executable UML.
This model of Executable UML is the so-called metamodel.
- What is meta?
 - Wikipedia:
 - 1) ...is a prefix used to indicate a concept which is an **abstraction** from another concept.
 - 2) ...the prefix meta- is used to mean **about** (its own category).
 - For example, metadata is data **about** data.
 - ...so a metamodel is a model **about** a model. 😊
- What does a metamodel do?
 - First of all it specifies the concepts of the language you are using to build a model, like a **class** or an **attribute**.
 - It also specifies what these concepts mean (rules), what you are allowed to do (or not), and it must be precise about it:
 - *"A class have exactly one name."*
 - *"A class can have zero or many attributes."*
 - *"The values an attribute can take is defined by exactly one datatype."*
 - At the same time as the metamodel defines the modelling language, it also defines the structure of a repository, where your models can be stored and accessed during model editing and model transformations.

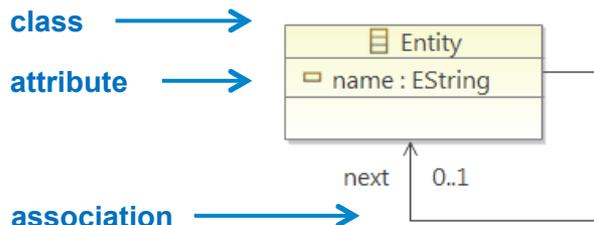
xtUML Metamodel



Metamodel – How can I build one?



- The primary tasks for a metamodel:
 - The metamodel defines how different concepts in a modelling language fit together.
 - **NOTE:** It also defines how different language concepts not can be combined.
 - It's all about organizing *pieces of data* in a precise way that can not be misinterpreted. (not by the users building models, and not by the users building model compilers)
- How a metamodel can be built:
 - UML-type metamodels are typically built (visualized) using UML Class Diagrams.
 - ...and in a Class Diagram the concept for a *piece of data* is a Class.
 - **"This is confusing!!! I build Class Diagrams in my Web Application Model ?"**
 - The thing is: The Class Diagram in a metamodel is just like any other Class Diagram.
 - There is no difference in the meaning of **class** for the class *WebPage* and the meta-class *Class*.
 - They are both just classes, and one just happens to be describing itself. (more on that later)
- Metamodels as Class Diagrams:
 - A metamodel is often a set of Class Diagrams. (typical for large problem domains)
 - ...where the key concepts are **Class**, **Attribute** and **Association**.
 - Building a metamodel is a process of abstracting out concepts as **classes**, identifying the properties of concepts as **attributes** and linking the concepts together across **associations**.



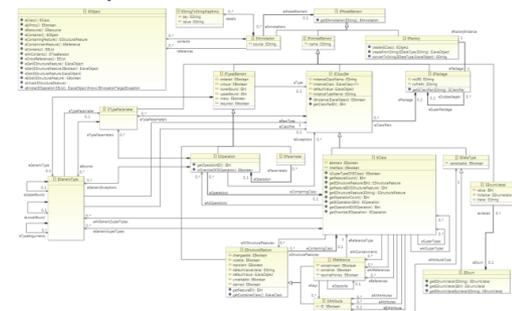
if still confusing:

almost like

A possible C code version

```
struct Entity
{
    char* name;
    struct Entity* next;
};
```

A complete metamodel - the Ecore



Metamodel – ...and for what ?

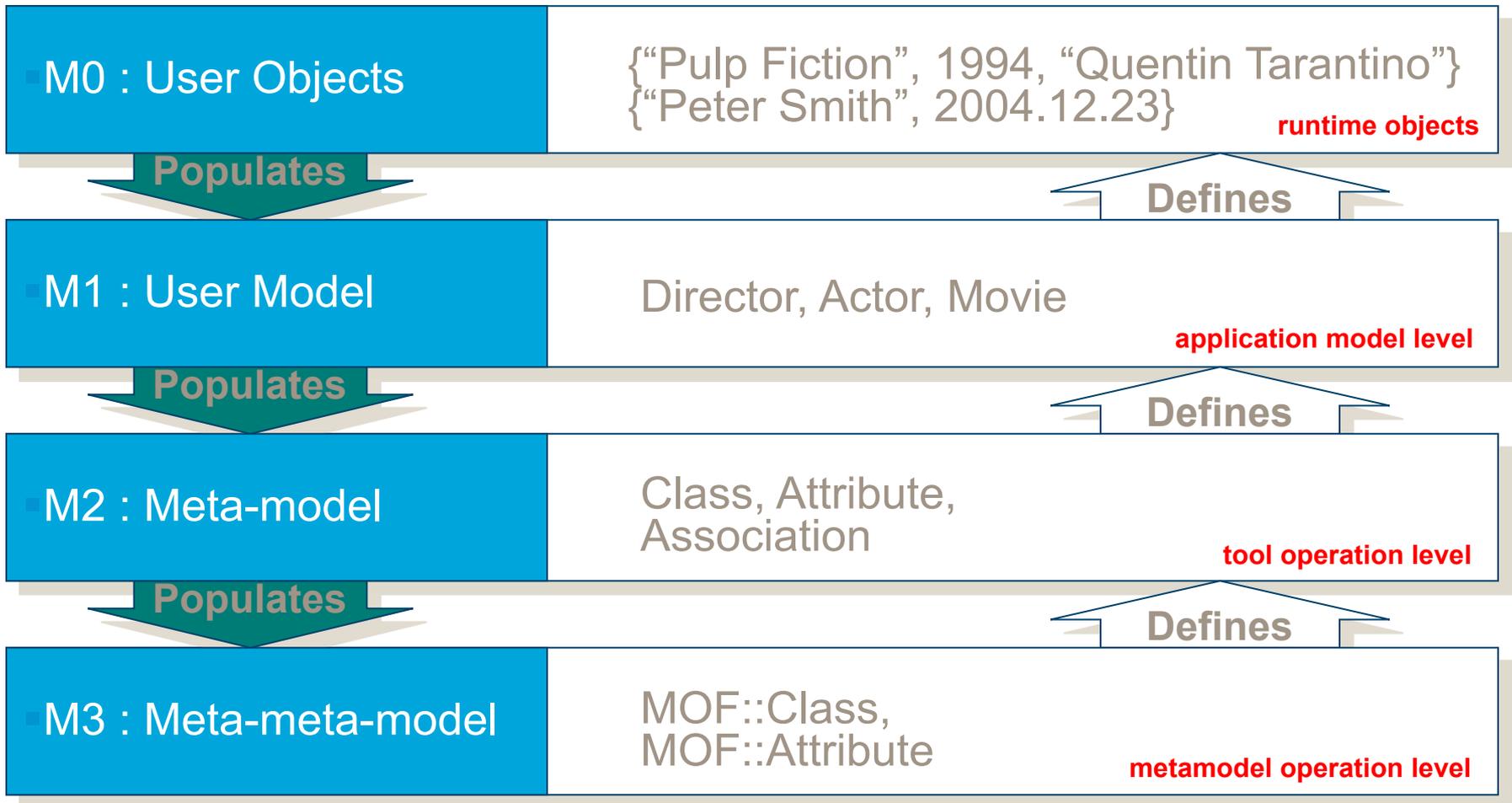


- Question: And for what can we build metamodels ?
 - Answer: Anything you can think of !!! 😊
 - 1) A programming language.
 - 2) An Action Language for UML. (as exists in Executable UML languages)
 - 3) Marking Model for code generation control.
 - 4) Deployment model for a target platform.
 - 5) ...and so on.
- What can we expect to find in different metamodels ?
 - Concepts (abstractions) in the metamodel are related to the nature of the problem domain.
 - Example:
Concepts in a programming language are quite different from concepts in a target platform.
 - ...so each metamodel has its own "vocabulary".
- Programming language examples:
 - Java Metamodel concepts: ***Class, Constructor, Interface***
 - ASM Metamodel concepts : ***Instruction, Register, Address***
 - UML Action Language Metamodel concepts : ***Signal, Port, Send, Link, Select***
- Platform examples:
 - Deployment Metamodel concepts: ***Priority, StackSize, MemoryPool***
 - PSM Metamodel concepts: ***Thread, Channel, Timer, Queue***

Metamodel – Meta layers



- The Meta Object Facility (MOF) is OMG's adopted standard for metamodeling.
 - MOF is the bottom level (M3) in a four-layer metamodel architecture.

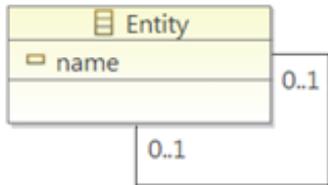


Metamodel – A repository view

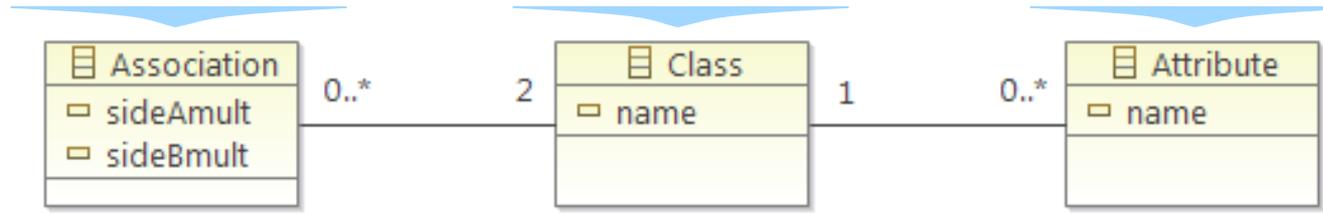
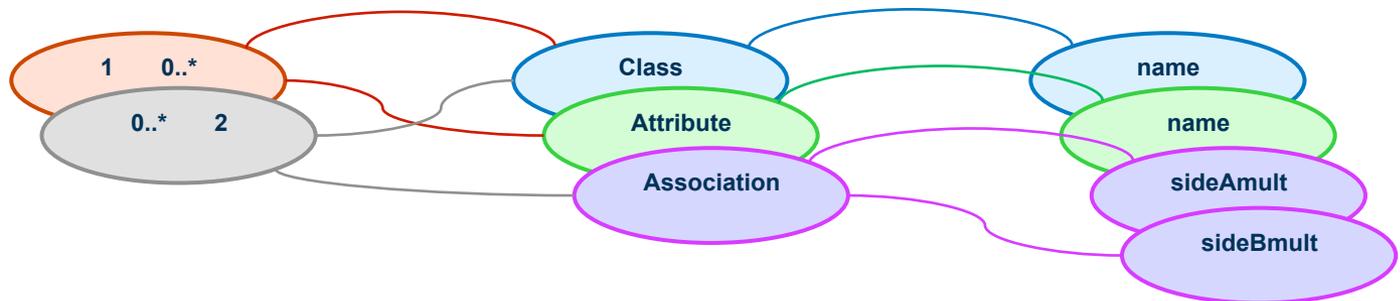


- How far up in level of abstraction can we go?
 - You have reached the highest meta-level of abstraction (M3) when the model describing the language is self-describing, such as the xtUML metamodel or the EMF metamodel named Ecore.
 - OK, but what can such a metamodel look like? (simplified example below)
- What does the repository for this metamodel look like? ...and how is the metamodel itself stored in it?
 - A metamodel is also just another model...
 - ...which can be stored as an instance of itself in a repository defined by itself. (illustrated immediately below)

A simple user model...



...modelled using concepts in a modelling language defined by a metamodel.



If we abstract out the concepts **Class**, **Attribute** and **Association** we get a metamodel looking like this:



In-memory Model Repository

A self describing metamodel, but also just another model...

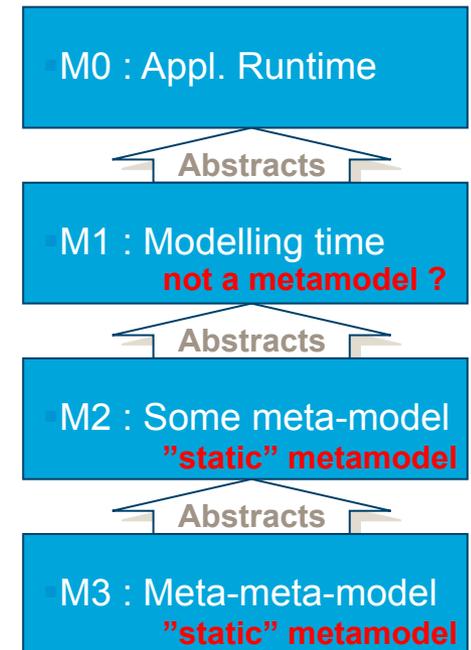
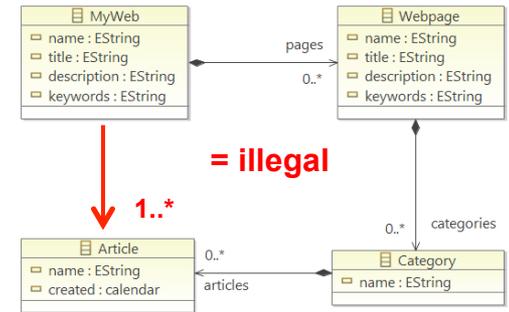
Metamodel – Is M0-M3 confusing?



- Which level is not a meta-level, really ???
 - Even a simple web-page application model can be thought of as a metamodel, which defines the rules for how objects of different concepts can be linked in runtime.
 - ...and it also clearly shows what **can not happen** while the application is running.
 - The metamodel for the modelling language defines what you can do in modelling-time, while the application model defines what is allowed to happen in run-time.
 - ...so in a sense, all models at any meta-level below M0 can be thought of as metamodels, or metamodels can be thought of as just ordinary models.
 - Precise modelling of data is (or should be) done in the same way in **any** model, guided by some development methodology. (= thinking process)

- Yes, M0-M3 levels and the "meta" word is a bit confusing...
 - One "meta" level is an **abstraction** of the level above.
 - ...so which level below M0 is not a meta-level ???
 - The thing is that M1 is "meta" for what can and can not happen at the M0-level runtime !!!
 - Need for demystify this "meta" even more? (...but how?)

EMF Tutorial model by Lars Vogel.





ERICSSON