

# Graphical DSLs

---



[www.mdsd-buch.de](http://www.mdsd-buch.de)



[www.mdsd-book.org](http://www.mdsd-book.org)

**Markus Völter**

[voelter@acm.org](mailto:voelter@acm.org)

[www.voelter.de](http://www.voelter.de)



Ingenieurbüro für Softwaretechnologie

[www.voelter.de](http://www.voelter.de)

- 1 -

© 2003 - 2006 Markus Völter

## UML Tools as DSL editors

- If you want to use a UML tool as the editor for your DSL, the DSL must be mapped onto a **UML profile**, i.e. a set of
  - Stereotypes,
  - Tagged values, and
  - Constraints
- **Pros:**
  - Easy to implement
  - (Looks like a) standard
  - More or less acceptable export format (XMI)
  - Model Management features (searching, partitioning...)
  - Integration into the tool suite (versioning, etc.), if provided by the UML tool and indeed necessary



Ingenieurbüro für Softwaretechnologie

[www.voelter.de](http://www.voelter.de)

- 2 -

© 2003 - 2006 Markus Völter

## UML Tools as DSL editors II

- **Cons:**
  - Graphical Features are very limited (especially, data-dependent graphics!)
  - Constraints cannot be checked in real time (this may change over time – tool vendors improve!)
  - You cannot „remove“ UML semantics completely (try to draw a line between two attributes...).
  - You always have to use the complete UML tool – complex UI

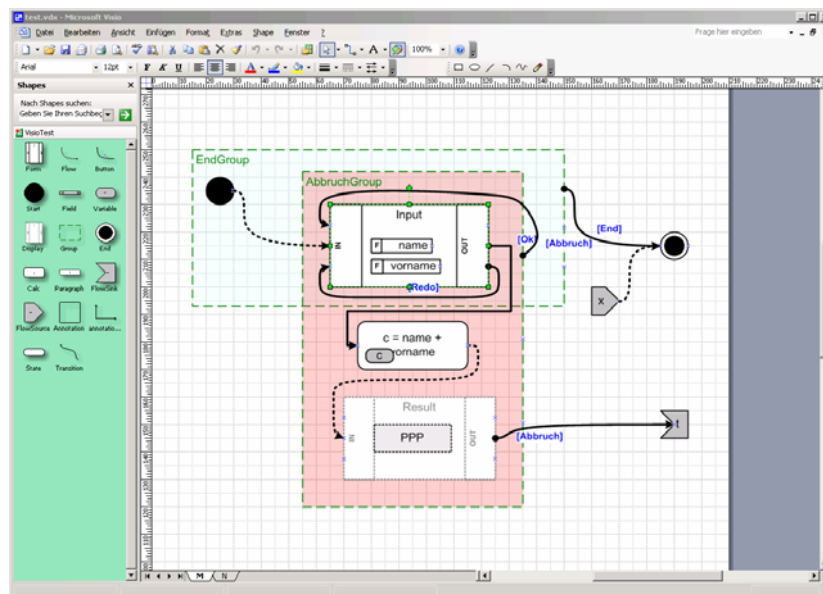
## Editor Alternatives: Adapting a Generic Drawing Tool

- If you want to use a generic drawing tool (such as Visio), a **connection** between the to-be-built model and the “drawing” must be made.
  - Export/Import Format
  - Metamodel Mapping
- **Pros:**
  - Implementation Effort Depends on Tool (Visio: easy!)
  - Powerful graphics features, Data Dependent Graphics possible (at least for Visio)
  - No inherited features forced upon you by the tool

## Editor Alternatives: Adapting a Generic Drawing Tool II

- **Cons:**
  - Usually, the tool knows nothing about the metamodel, i.e.
    - Very limited concrete syntax checking
    - No realtime constraint checking
  - A later phase (after saving) validated syntax and constraints
  - Often limited partitioning and model management support
  - Proprietary – no standard.

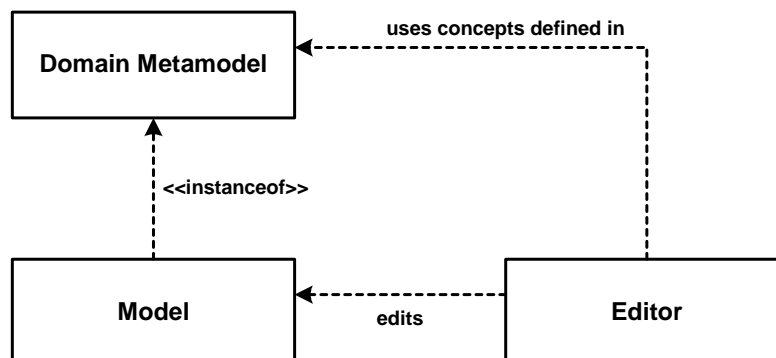
## Editor Alternatives: Adapting a Generic Drawing Tool III



## Editor Alternatives: Developing a Custom Tool

- **Pros:**
  - Typically, quite powerful graphics features
  - No inherited features forced upon you by the tool
  - Only valid concrete syntax is possible
  - Constraints can be checked in realtime
  - Tool can be as simple as possible – no UML mess
- **Cons:**
  - Implementation Effort usually high
  - Often limited partitioning and model management support (depends ultimately on the effort you want to put in)

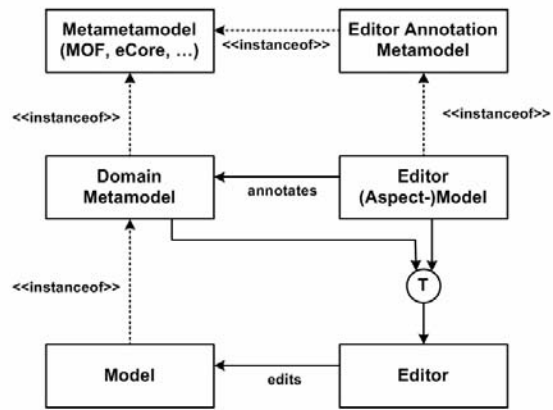
## Editor Alternatives: Developing a Custom Tool II



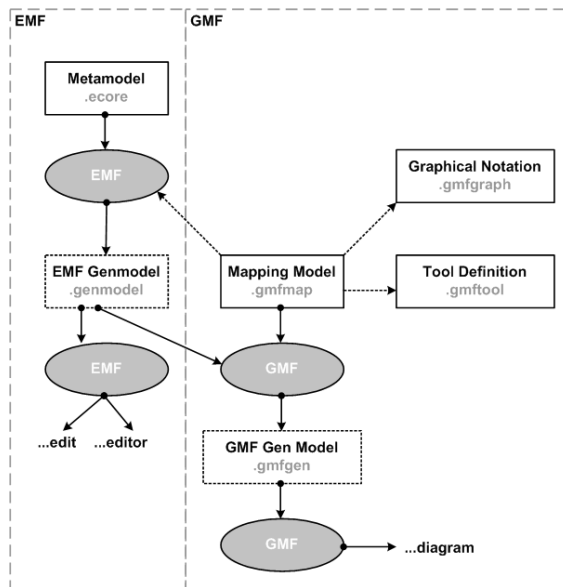
- Models are instances of their metamodel.
- The Editor edits models...
- ... and uses concepts from the metamodel

## Editor Alternatives: Custom Tool / Generating Editors

- GEF (Eclipse Graphical Editing Framework) is powerful, but hard to use.
- Especially, if the domain metamodel changes regularly in a project, adapting the editors correspondingly is annoying.
- As a consequence, we **generate the editors** from the domain metamodel and additional editor descriptions.



## GMF Process



# Custom Tool – the generated Network Editor

