

# Engineering Modeling Languages

(Metamodeling + XMI)

Dániel Varró

Foundations of Model Driven System Development

## Language-Driven Development: Why?

- Challenge of today's SW development
  - Complexity:
    - increased devel time and costs
  - Diversity:
    - domains, requirements, implementation techniques, tools
  - Change: „The only constant is change”
    - Requirements, impl. techniques, bug fixes, etc.

## Language-Driven Development

- Languages are fundamental for communication between humans
- They should provide support for
  - Execution, Analysis, Testing, Visualization, Parsing, Translation, Integration
- Key: Abstraction + Integration of many languages
  - Transformation
  - Aspect integration
  - Synchronization
  - Verification of Refinement/Equivalence
  - Evolution

## Evaluation of UML

- Advantages:
  - Standard common language
  - Visual notation
- Disadvantages:
  - Imprecise semantics
    - No single, integrated solution for all UML artifacts
  - Limited scope and flexibility
    - UML Profiles (Stereotypes)
  - Unified (NOT Universal) Modeling Language

## Metamodeling: The Design of Modeling Languages

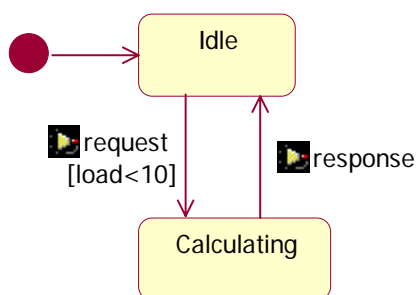
### Designing Modeling Languages

- Metamodeling: Design methodology of modeling languages
- Metamodel: model of a modeling language
- Features:
  - Concrete syntax
  - Abstract syntax
  - Well-formedness rules
  - Semantics
  - Mappings to other languages

## Concrete syntax

- The way models (or programs) are presented
  - Textual notation:
    - + Easy to write: Able to capture complex expressions
    - Difficult to read: Difficult to comprehend and manage after certain complexity
  - Visual notation:
    - + Easy to read: Able to express (selected / subset of) details in an intuitive, understandable form
    - + Safe to write: Able to construct syntactically correct models
    - Difficult to write: graphical editing is slower

## Example: Concrete Syntax



```
void request() {
    if (state == "idle" &&
        this.load < 10)
        state = "calculating";
}

void response() {
    if (state == "calculating")
        state = "idle";
}
```

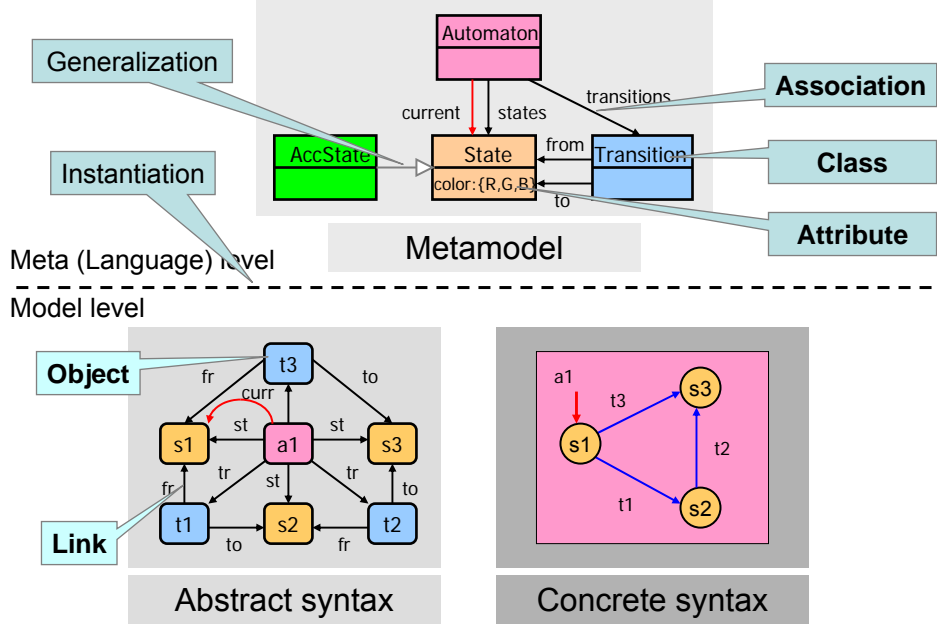
Graphical notation

Textual notation

# Abstract syntax (Metamodel)

- Metamodel: a model of a modeling language
  - Means: above, transcending
- Goal: to define
  - The vocabulary of concepts in the language
  - How these concepts can be combined to create models
- Contents:
  - Definition of concepts
  - Relationships between these concepts
  - Abstraction/Specialization (Taxonomy, Ontology) between the concepts
  - Constraints (e.g. Multiplicity)
  - (Additional well-formedness rules)

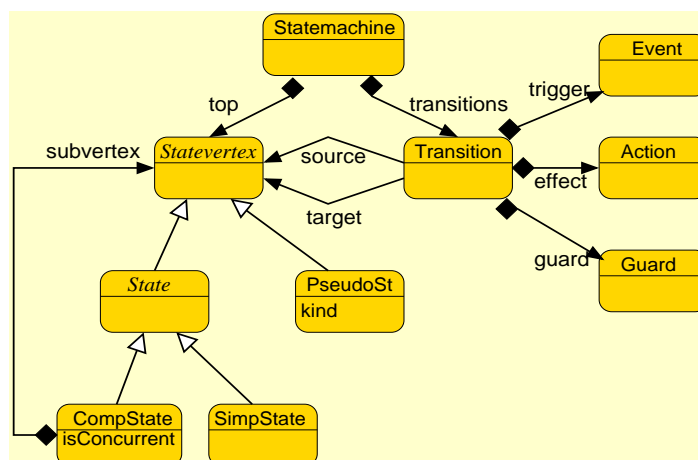
## Metamodels and model instances



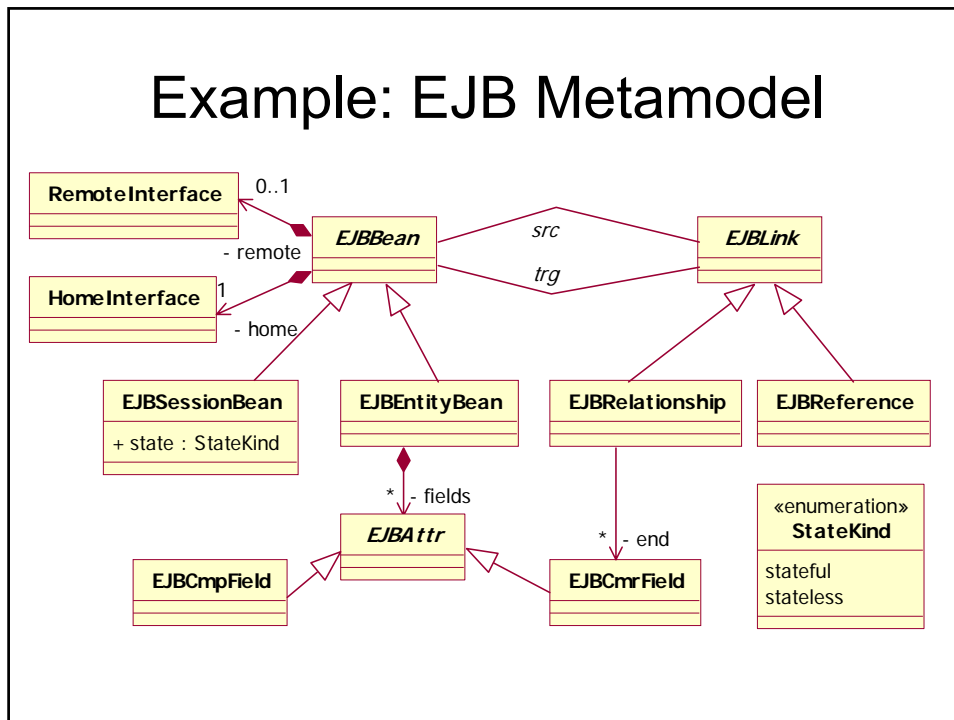
## Type conforming models



## Example: Statechart Metamodel



## Example: EJB Metamodel



## Where do you find metamodels?

- Different application domains around UML
  - SysML (systems engineering)
  - SPEM (process modeling)
  - CWM (data warehousing)
  - GRM (Profile for Schedulability, Performance and Time)
  - EDOC (enterprise modeling)

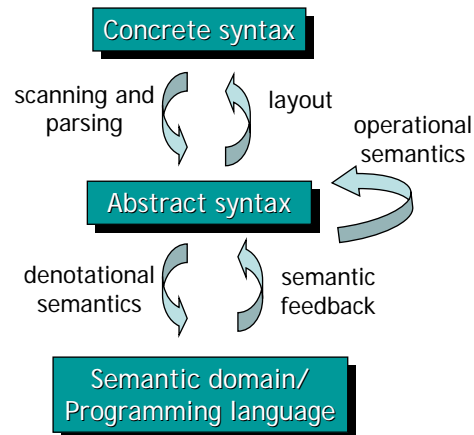
## Well-formedness rules

- Multiplicity constraints
  - At most one: 0..1
  - Many: \*
- Aggregation/Containment
  - At most one parent for each model element
- Language specific constraints:
  - E.g.: Each state in a class must have a unique name
  - Expressed in e.g. OCL

## Dynamic Semantics

- Semantics: the meaning of concepts in a language
- Main approaches:
  - Denotational (Translational): translating concepts in one language to another language (called semantic domain)
    - compiled
  - Operational: modeling the operational behavior of language concepts
    - interpreted

# Overview of Transformations



**XMI:**  
Document design with metamodels

## Dokumentumtervezés metamodellezéssel

- **Célkitűzés:**
  - XML alapú dokumentumok struktúrájának automatikus szintézse
  - az XMI (XML Metadata Interchange) szabvány használatával

## XML áttekintés

- **XML = eXtensible Markup Language**
  - a Web szabványos nyelve
  - strukturált információátvitel
- **Korábbi problémák:**
  - HTML: megjelenítés orientált
  - SGML: túl komplex
- **Tulajdonságai:**
  - különválasztott struktúra és megjelenítés
  - eszköz- és gyártófüggetlenség

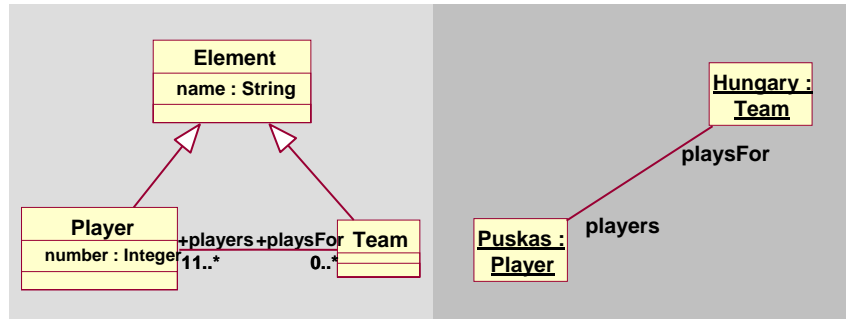
## Az XML struktúrája

- XML Schema, DTD (Document Type Definition):
  - dokumentumok modellezésére
  - speciális nyelvtan
  - Dokumentumellenőrzés
- Dokumentum példány:
  - információ tárolására
  - jól formált HTML-szerű tag-ek
  - szigorú fastruktúra

## Dokumentum definíciók és példányok

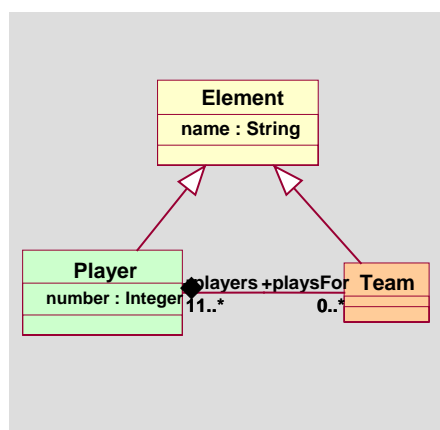
	Definition	Instance
<b>element</b>	<!ELEMENT Car (Model, Price?,Extras*)>	<Car> <Model>Mondeo</Model/> </Car>
<b>attribute</b>	<!ATTLIST Car owner CDATA>	<Car owner="John">

## Példa: metamodell és modell



- Csapat metamodell
  - M2 szint
- Csapat modell
  - M1 szint

## Példa: XMI 1.0 DTD

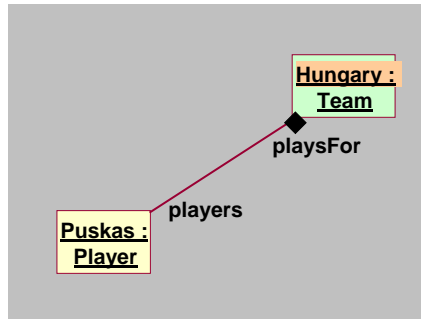


```

<!ELEMENT Team.players (Player)*>
<!ELEMENT Player.playsFor (Team)*>
<!ELEMENT Element.name
    (#PCDATA|XML.reference)* >
<!ELEMENT Team (Element.name,
    XML.extension*,
    Team.player) >
<!ATTLIST Team
    %XML.element.att
    %XML.link.att >
<!ELEMENT Player (Element.name,
    XML.extension*,
    Team.playsFor) >
<!ATTLIST Player
    %XML.element.att
    %XML.link.att >
    
```

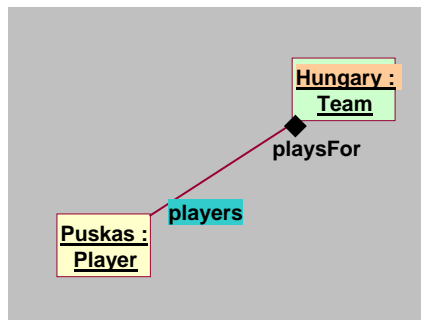
## Példa: XMI 1.0 dokumentum

```
<Team id='t1'>
  <Element.name>
    Hungary
  </Element.name>
  <Team.players>
    <Player id='p1'>
      <Element.name> Puskas
    </Element.name>
    <Player.number> 10
    </Player.number>
    <Player.playsFor
      xmi.idref='t1' />
    </Player>
  </Team.players>
</Team>
```



## Példa: XMI 1.1 dokumentum

```
<FB:Team id='t1'
  name='Hungary'>
  <FB:Team.players>
    <FB:Player id='p1'
      name='Puskas'
      number='10'
      playsFor='t1' />
    </FB:Player>
  </FB:Team.players>
</FB:Team>
```



## Példa: XMI 2.0 dokumentum

```
<fb:Model xmlns:fb="...",
  xmlns:xmi="..."
  <teams xmi.type="Team"
    xmi.id="t1"
    name="Hungary">
  <players xmi.id='p1'
    name='Puskas'
    number='10'
    playsFor='t1' />
  </teams>
</fb:Model>
```

