Modelling and Analysing the Behaviour of Software

James Williams
University of York

Overview

- Model-driven engineering
    - Model management : Epsilon
- Integrating formal methods with MDE
    - Model Behaviour Language

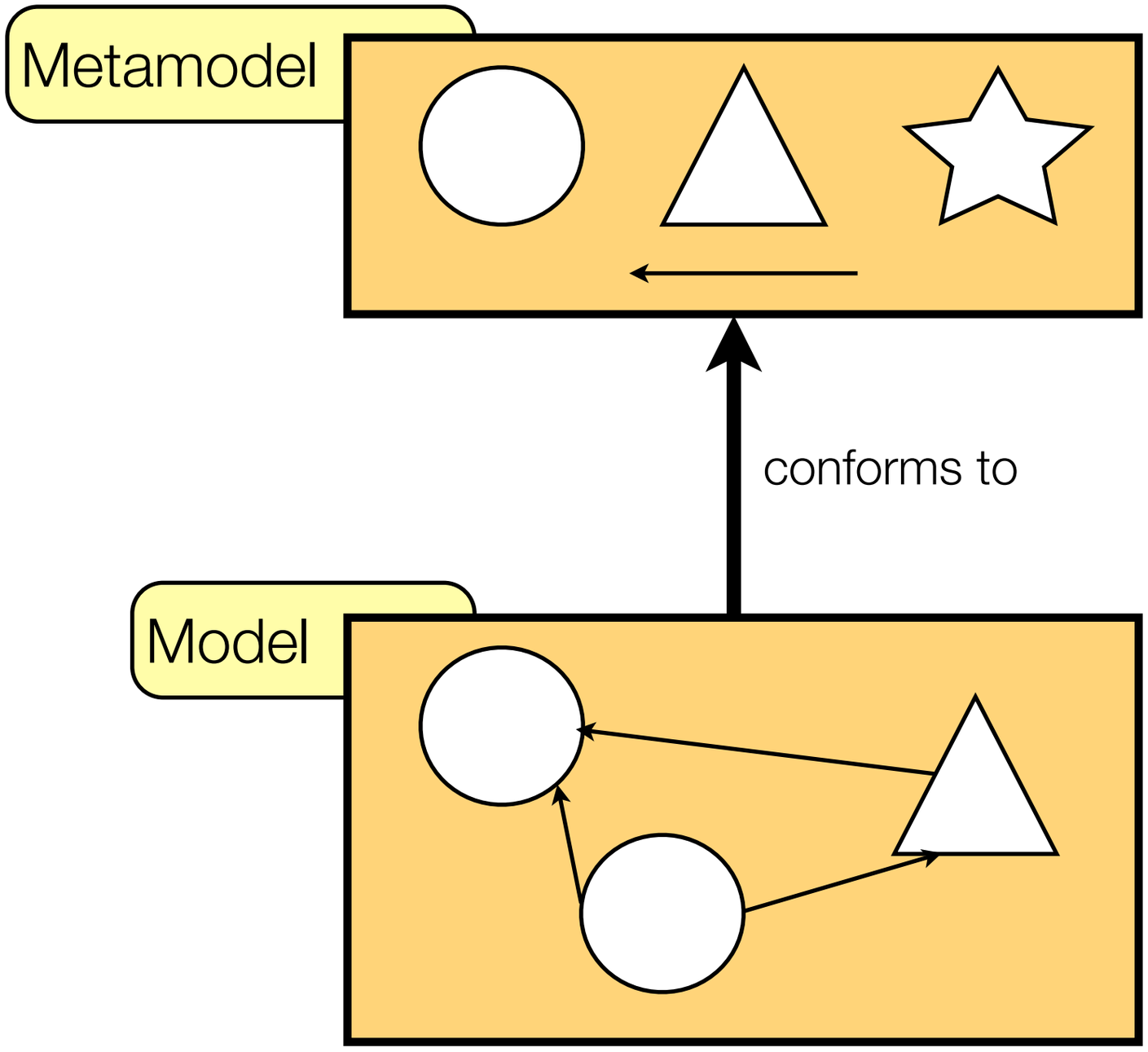
Model-Driven Engineering

Overview

Model-Driven Engineering

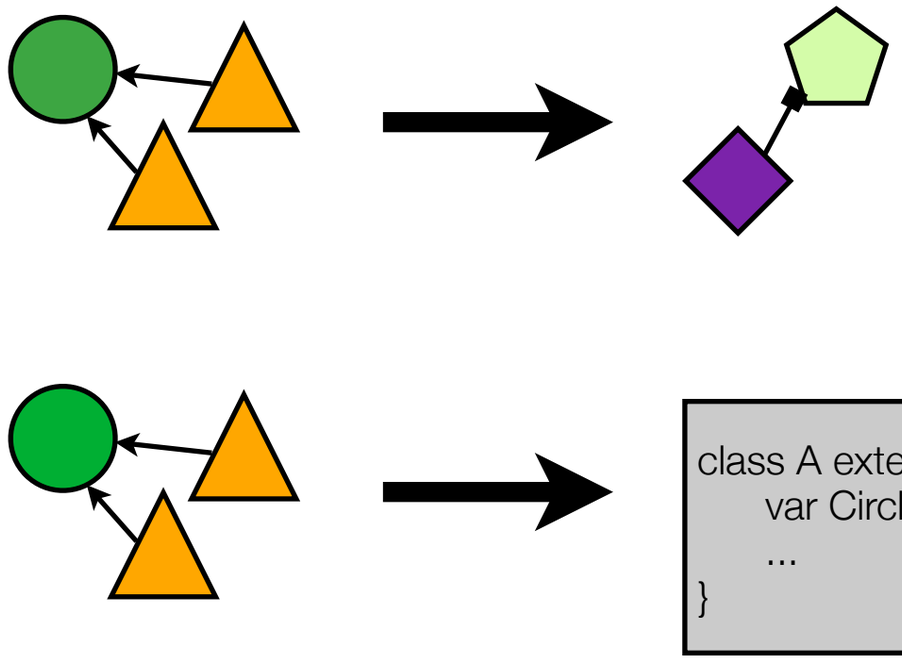
- Promotes models to first-class artefacts
- More than documentation
- Live entities that are amenable to automated processing

MDE: Metamodels and models



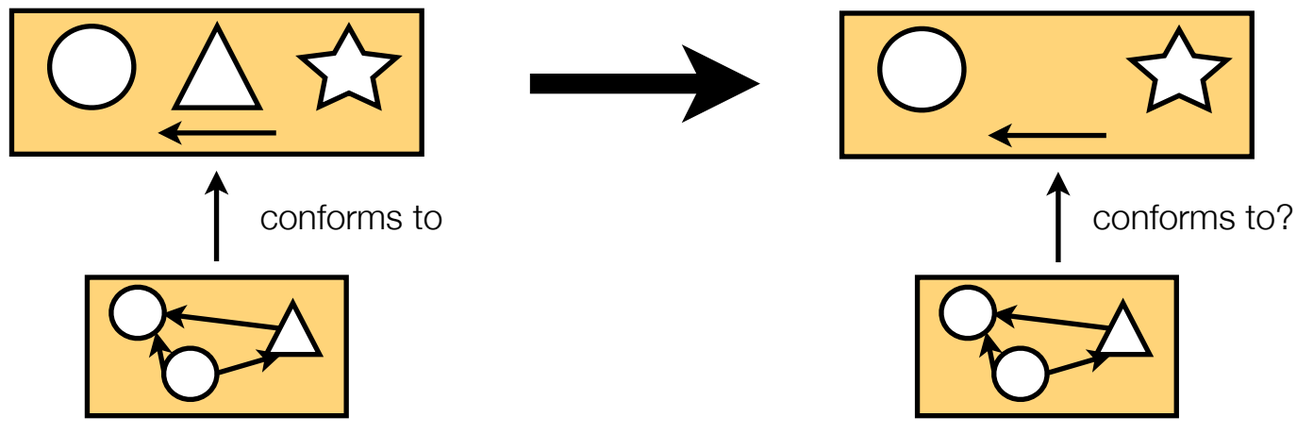
MDE: Model management

- Transformation
  - Model-to-model
  - Model-to-text



```
class A extends Triangle {
    var Circle circle;
    ...
}
```

- Evolution
  - Model migration



- Merging, validation, comparison, ...

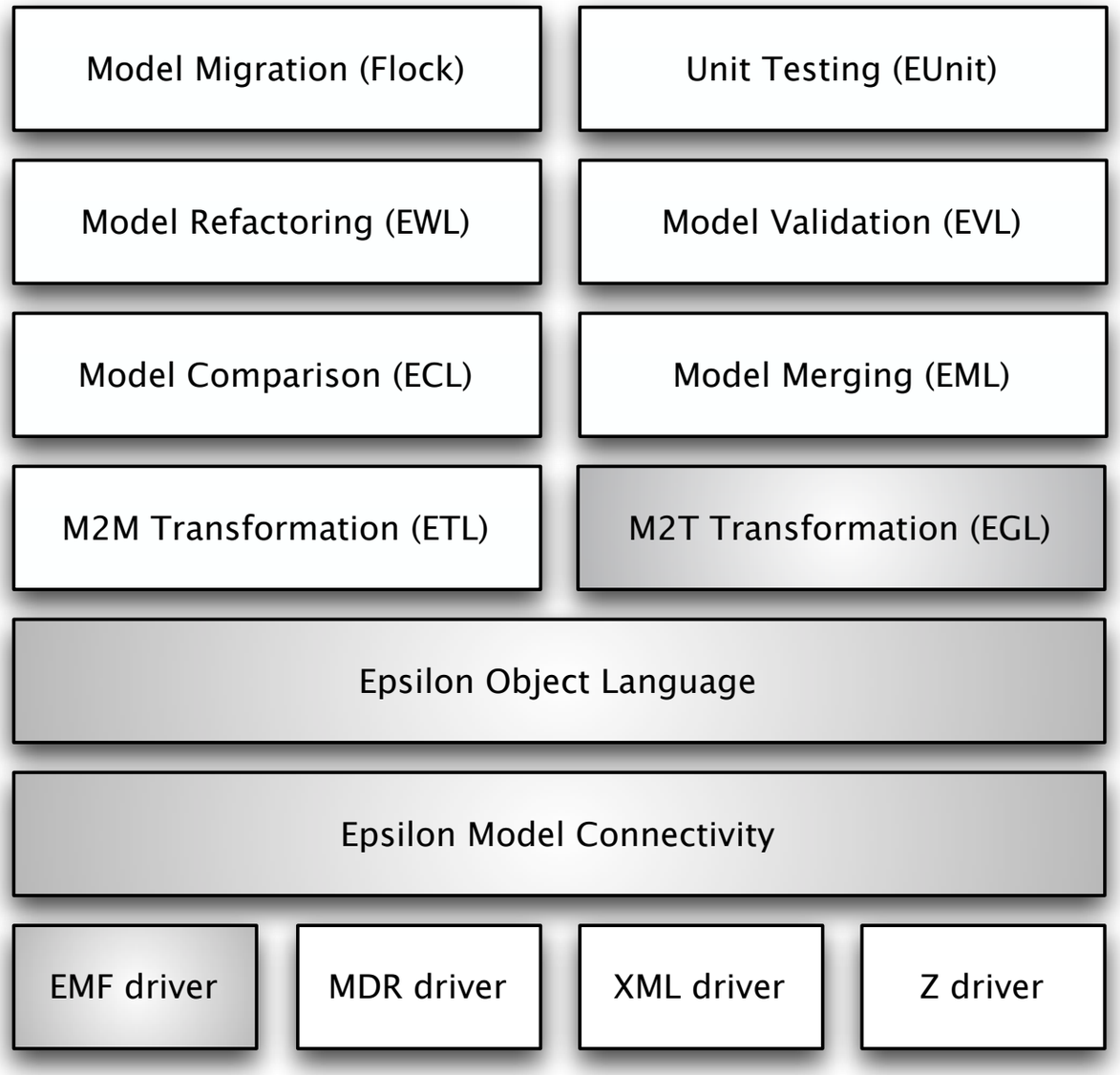
MDE: Models ≠ UML diagrams

- UML is just one modelling language
- Most domains have different abstractions/semantics
    - Domain Specific Languages (DSLs)
- Models ≠ Pictures
    - Models can be graphical or textual
    - ... or both

Epsilon

www.eclipse.org/gmt/epsilon

- A family of integrated languages and tools for MDE



Epsilon : More...

- Other languages for:
  - Model refactoring, model merging, model comparison, model validation, model migration
- Epsilon tools:
  - EuGENia : generates graphical editors from simple definitions
  - HUTN : implementation of the OMG’s Human Usable Textual Notation
  - Concordance : indexes and monitors models for reconciliation and reporting
  - ModeLink : managing cross-model references

Integrated Methods

MDE + Formal Methods

MDE vs Formal Methods

- MDE
  - Intuitive, graphical notations. Good for abstraction, composition
  - No support for *formally* analysing models - notations often lack rigorous semantics
    - Can be constrained using OCL, but cannot describe the complete dynamic behaviour of a class diagram
- Formal methods
  - Unambiguous semantics and are amenable to verification
  - But: high entry cost, “expert” knowledge required, viewed as being inaccessible

Integrating formal methods

- MDE can benefit from formal analysis
- Formal methods can benefit from graphical notations, good tool support

- “Semi-formal” models often transformed into a formal language for analysis
  - e.g. xUML --> PROMELA, UML --> Z [Ama07], UML --> Alloy [Ana07]
  - Issues: traceability, transformation validation, inefficient representation

Model Behaviour Language

Overview

The Model Behaviour Language

- A language for the specification and analysis of the structure and behaviour of models

- Integrated with the state-of-the-art practices in MDE
  - Model transformation
  - Code generation
  - Test-case generation

- In its infancy..!

MBL: Language overview

- Composed of three sub-languages:
    - State description language
    - Event language
    - Expression language

MBL: State Description Language

- Describes the *structure* of the model
- Mimics Emfatic notation (http://wiki.eclipse.org/Emfatic)
  - Textual language for describing **Ecore** models

```
package <name> {

  (class <name> {
    (attr <type> <name>;)*
    (val <type> <name>;)*
    (ref <type> <name>;)*
  })*
}
```

```
package ATC {
  class Aircraft { }

  class Airport {
    attr String name;
    val Aircraft[*] landed;
    ref Aircraft[*] permission;
  }
}
```

MBL: Event and Expression Languages

- Adds explicit *behavioural* definitions to the model

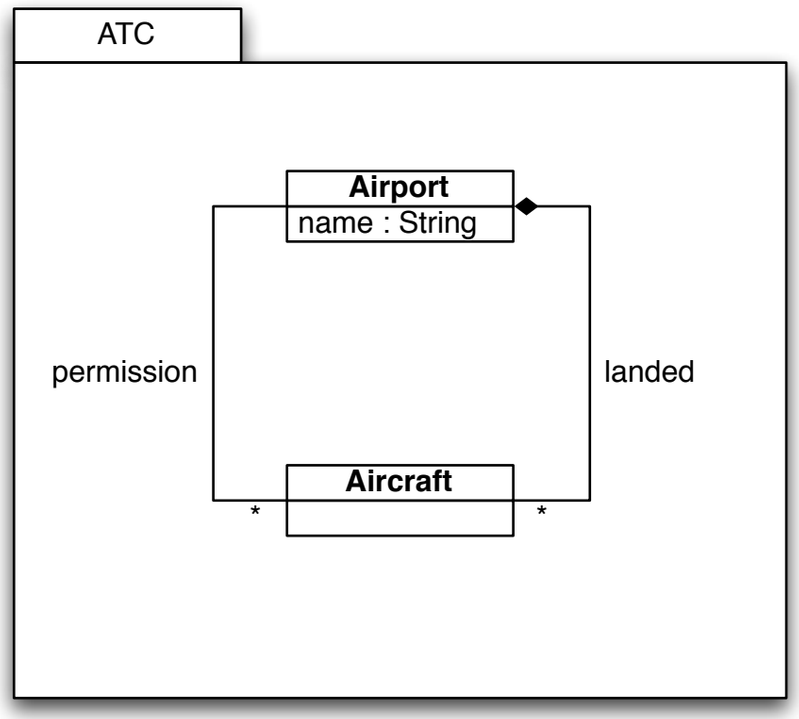
- Precondition, postcondition and body definitions

```
package <name> {
  ( class <name> {
    ( attr <type> <name>; )*
    ( val <type> <name>; )*
    ( ref <type> <name>; )*

    ( event <name> <param_list> ( [ <pre_conditions> ] )? {
      <event_body>
    } ( [ <post_conditions> ] )? )*
  } )*
}
```

MBL: Complete Example

```
package ATC {
  class Aircraft { }

  class Airport {
    attr String name;
    val Aircraft[*] landed;
    ref Aircraft[*] permission;

    inv RestrictNumberLanded: landed.size() <= 20;

    event givePermissionToLand(Aircraft a) [  ] {
      permission.add(a);
    }

    event landPlane(Aircraft a) [ permission.includes(a) ]{
      landed.add(a);
    }
  }
}
```

MBL: Complete Example (Diagrammatically)

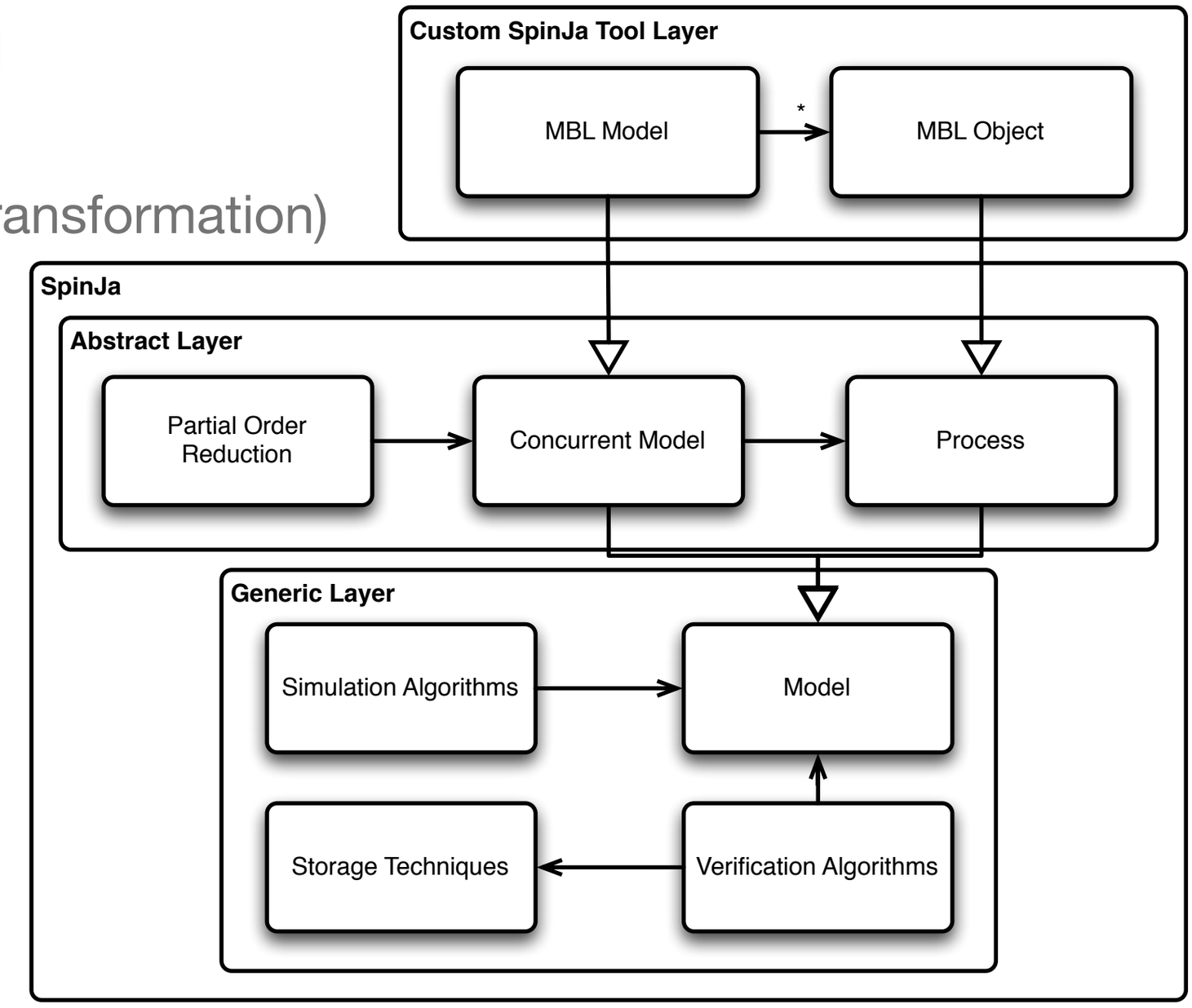


MBL: Events

- Events are abstract representations of actions that can occur
- Do not directly map to the model's structure
- Can be *refined* by operations

MBL: Analysis

- Events are the *transitions* between states
- Initial state specified in HUTN
- Model analysed directly (no transformation)
- Custom model checker built atop SpinJa
- Expressions evaluated using EOL



MBL: Future work

- Refinement: data and event
- Event-trace reasoning: temporal expressions
- Test case generation
- Constraints: generate EVL constraints and attach to model
- Code generation
- Interactive simulator

MBL: The Big Picture

- A language to formally specify the behaviour of MDE models
- Tooling to analyse (model check, simulate) the models
- Integrated into EMF - the de facto MDE technology
- Amenable to MDE tasks - e.g. transformations
- Code and test generation, model validation

Conclusions

- Model-driven engineering
    - Models treated as first-class citizens
- Epsilon: state-of-the-art platform for model management
- The Model Behaviour Language
    - Extends structural definitions of models to include behaviour specification
    - Analysis of behavioural models