

Object-Oriented Theories for Model Driven Architecture

Tony Clark, King's College, UK.

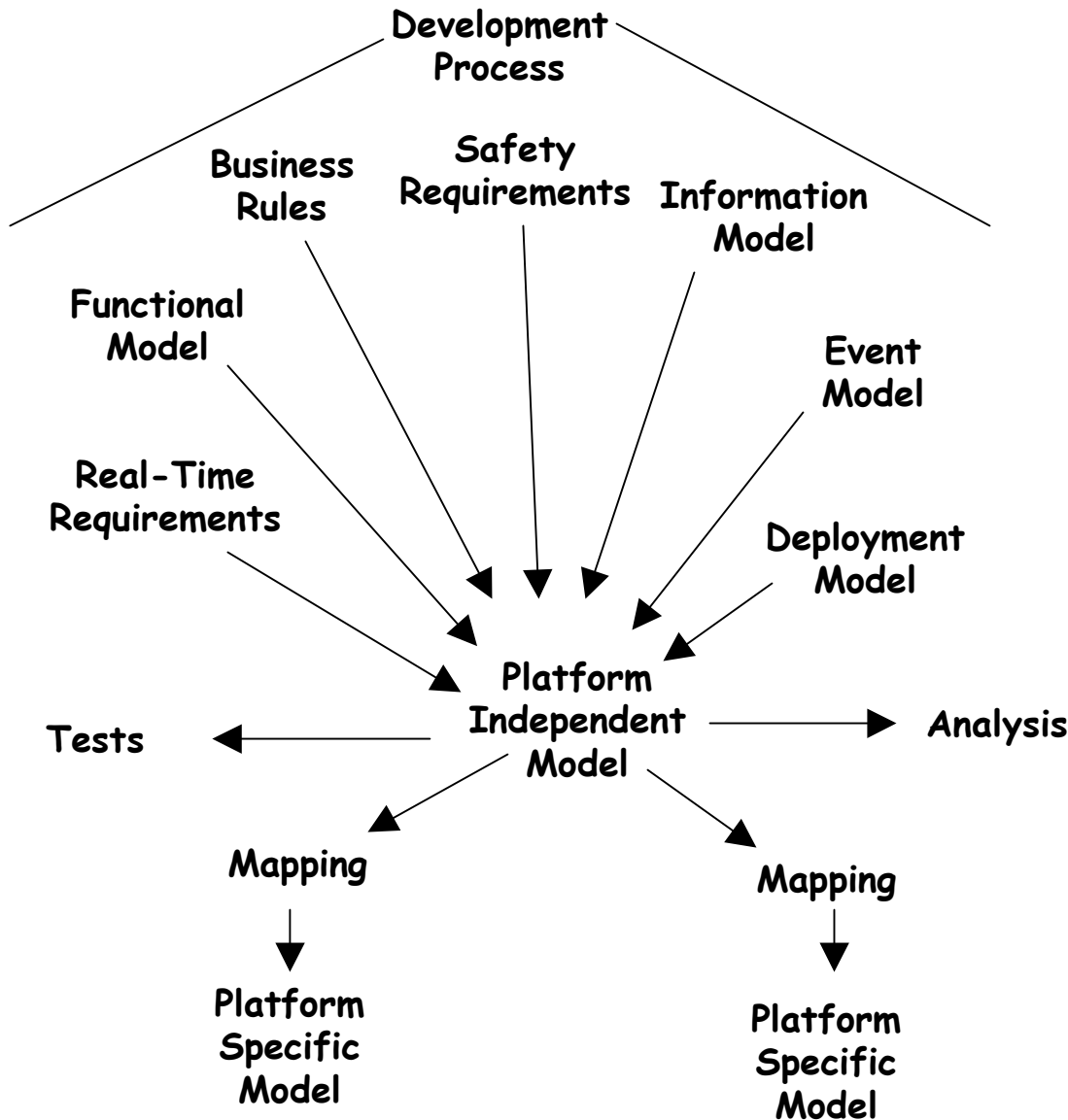
Andy Evans, University of York, UK.

Robert France, Colorado University, USA.

Overview

- Requirements for MDA
- Language Engineering for MDA
- Technology for MDA:
 - Package Extension
 - Templates
 - Theories
 - Refinement and Refactoring

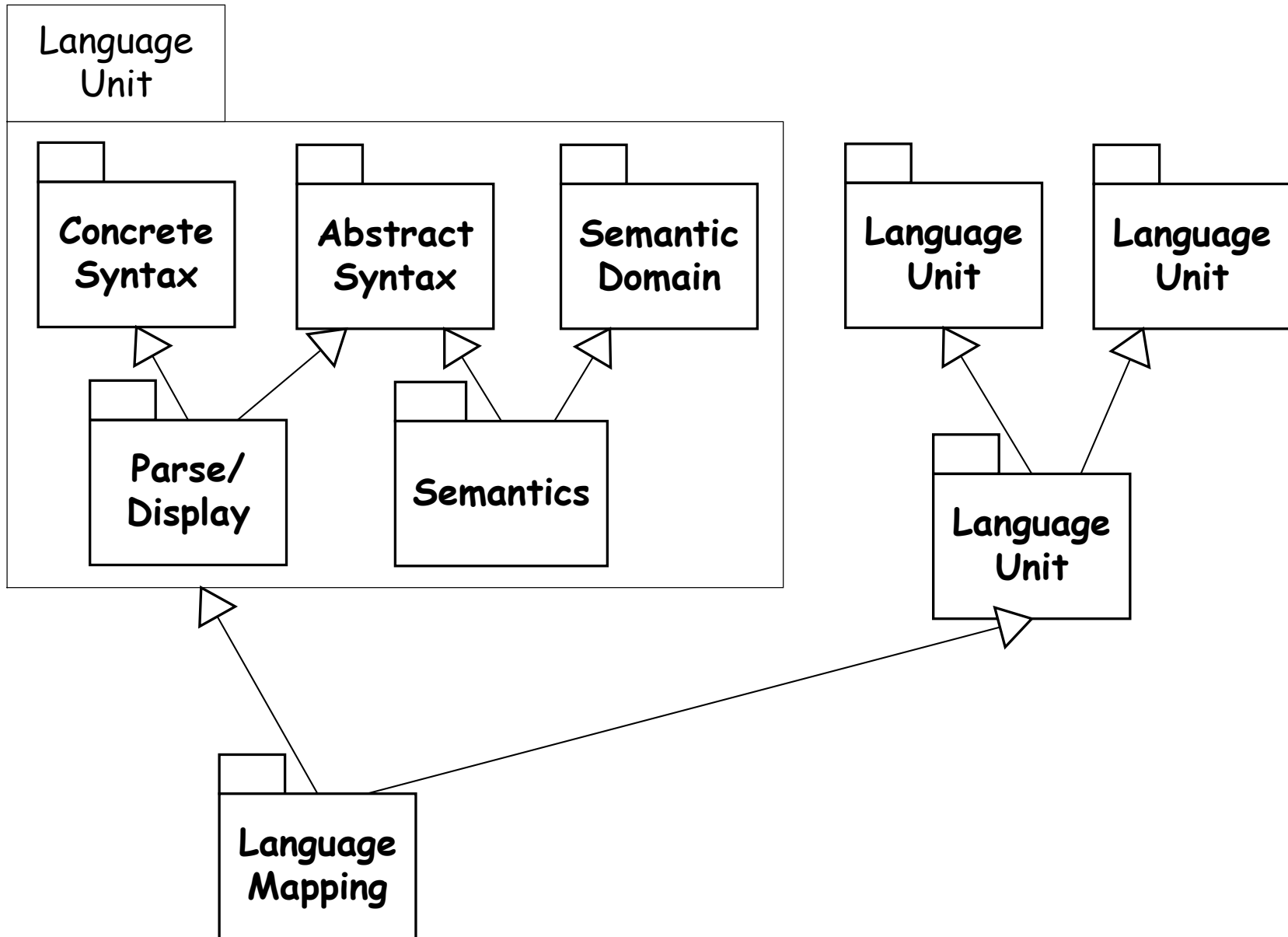
Model Driven Software Engineering



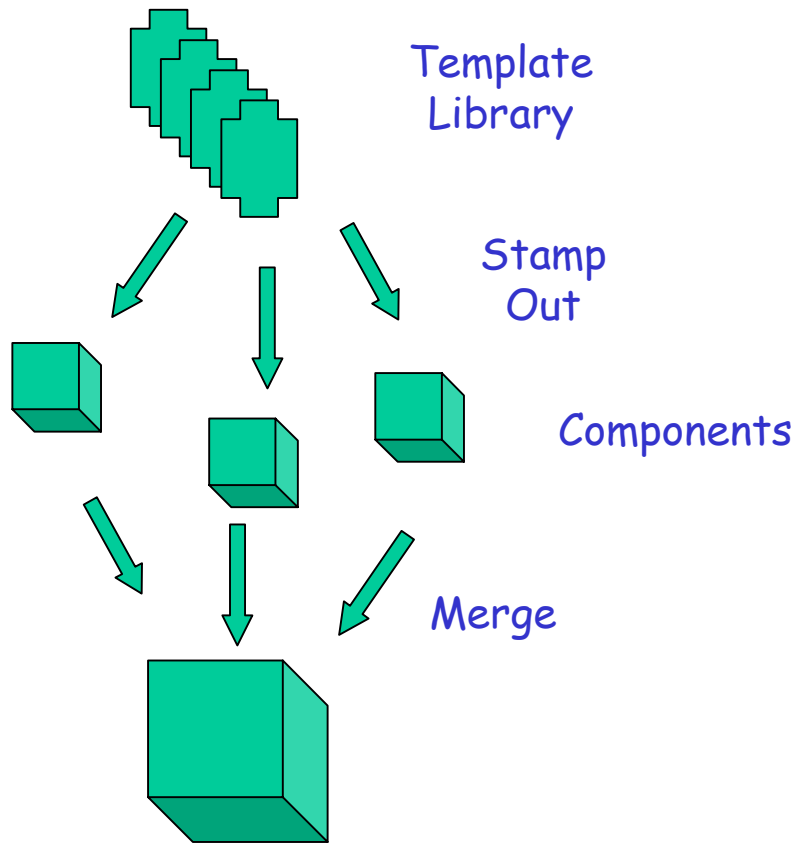
Software systems:

- increasingly modelled
- need reuse of design/code
- require tool support
- faster delivery times
- skills shortage
- product lines
- more automation
- interoperability
- domain specific languages

Language Engineering for MDA

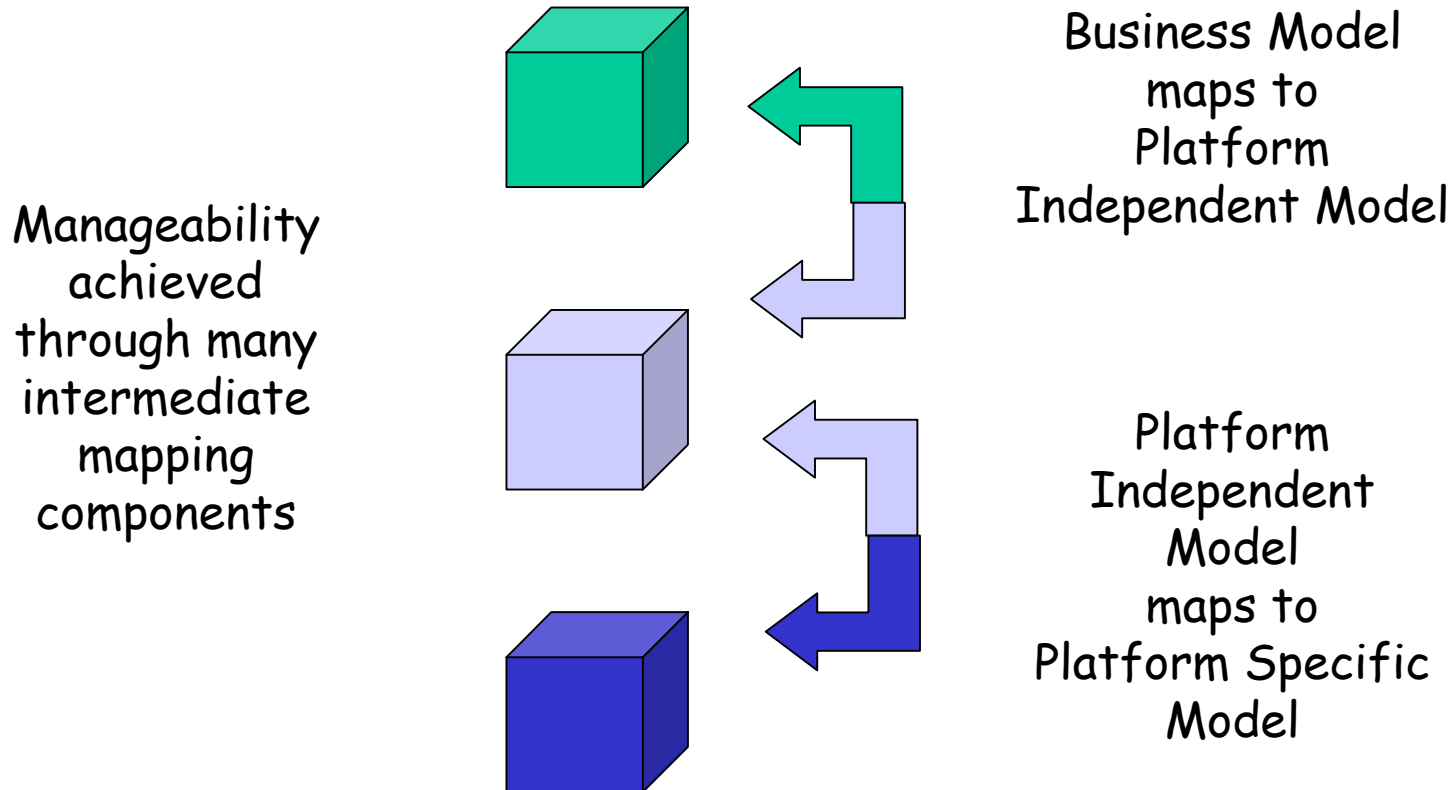


Template Packages and Classes

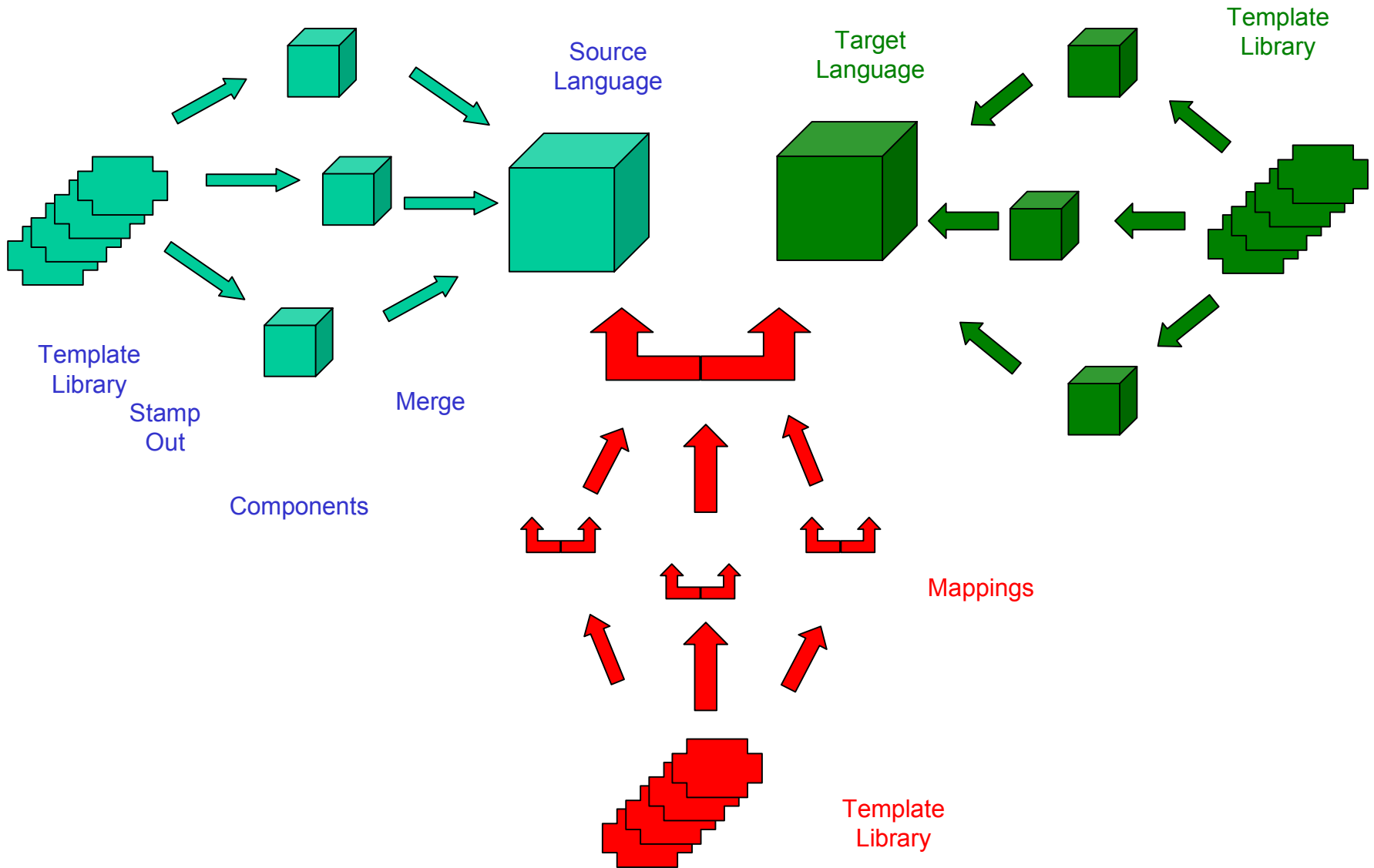


- MML allows definitions to be parametric.
- OO Patterns can be defined as templates.
- Templates are *stamped out*.
- Templates can be combined.
- Combination occurs through package extension.

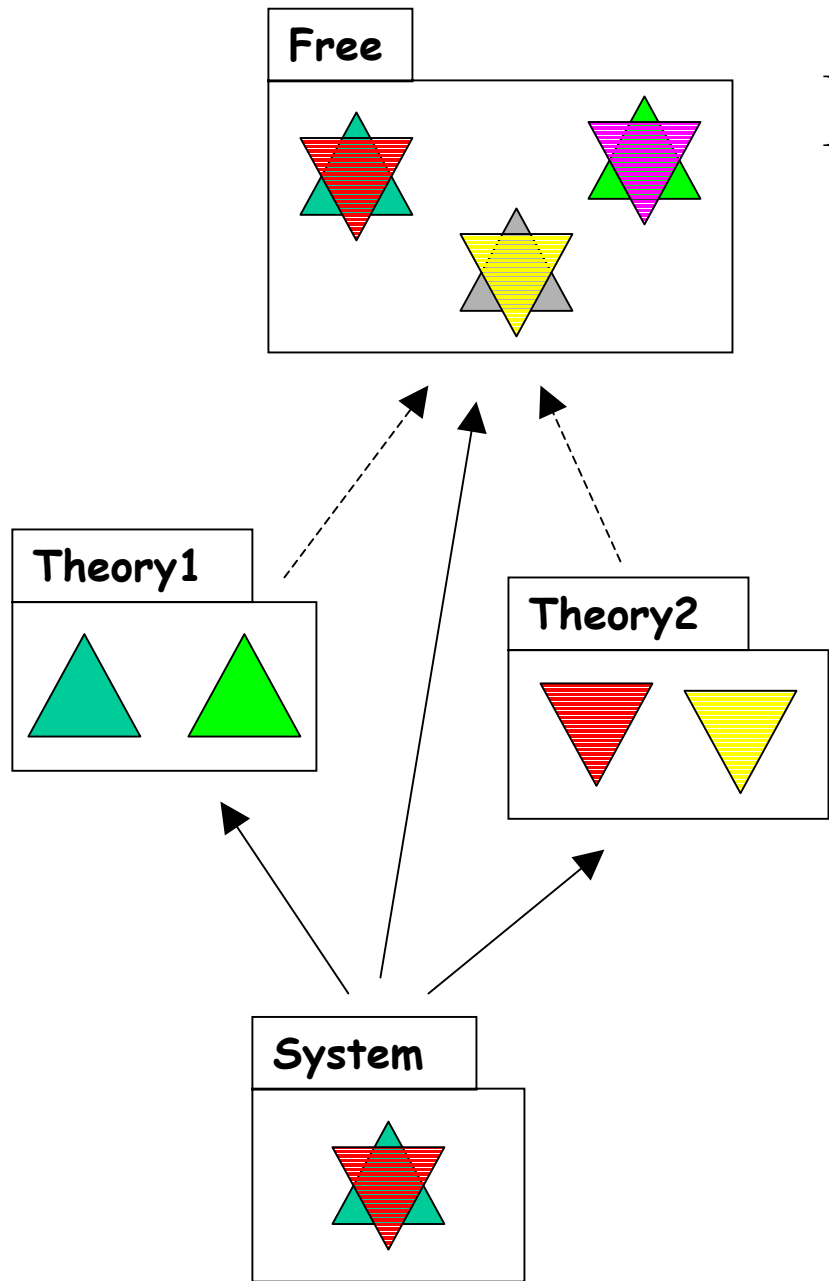
Realizing MDA With Mappings



Mappings for Component Based MDA



Modelling Theories for MDA



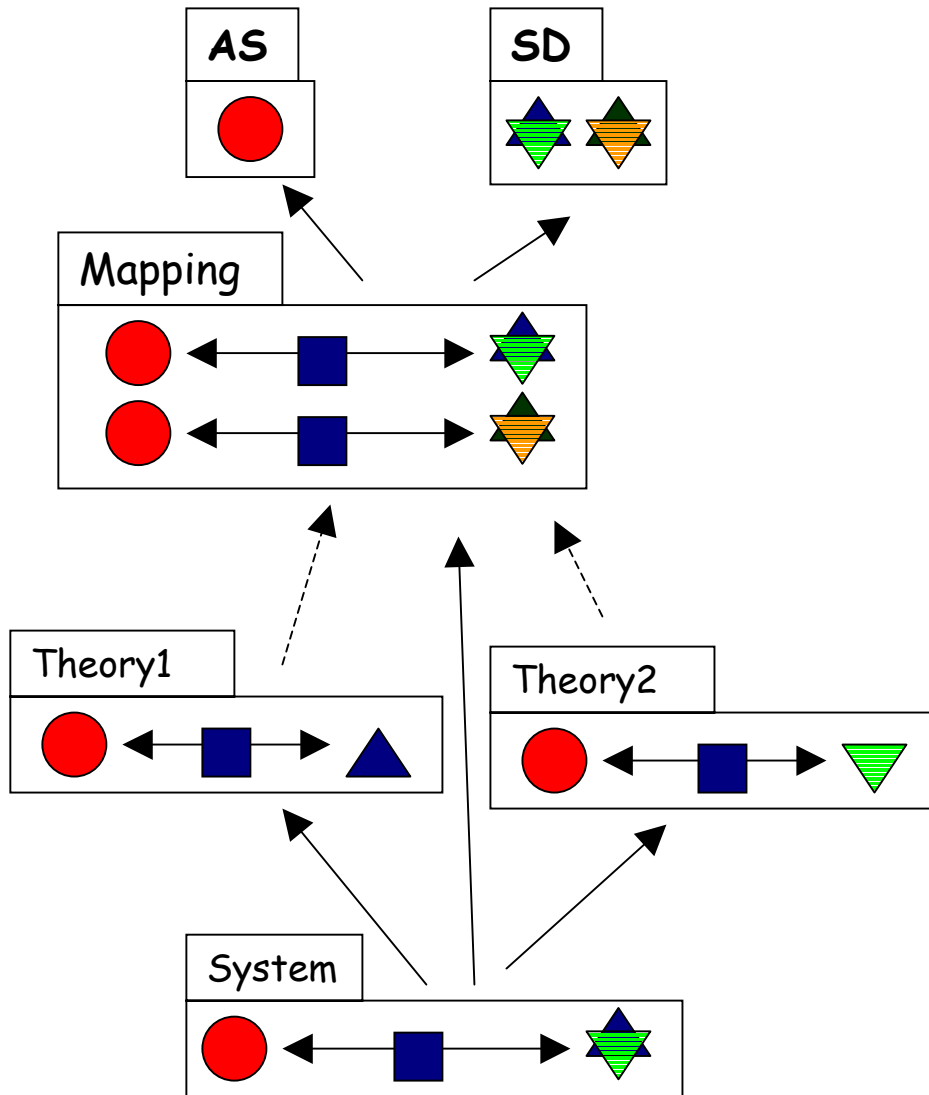
Models are freely constructed.

Theories are expressed as constraints on different system aspects.

Theories are partial views of a system.

Theories are combined to produce the final system description.

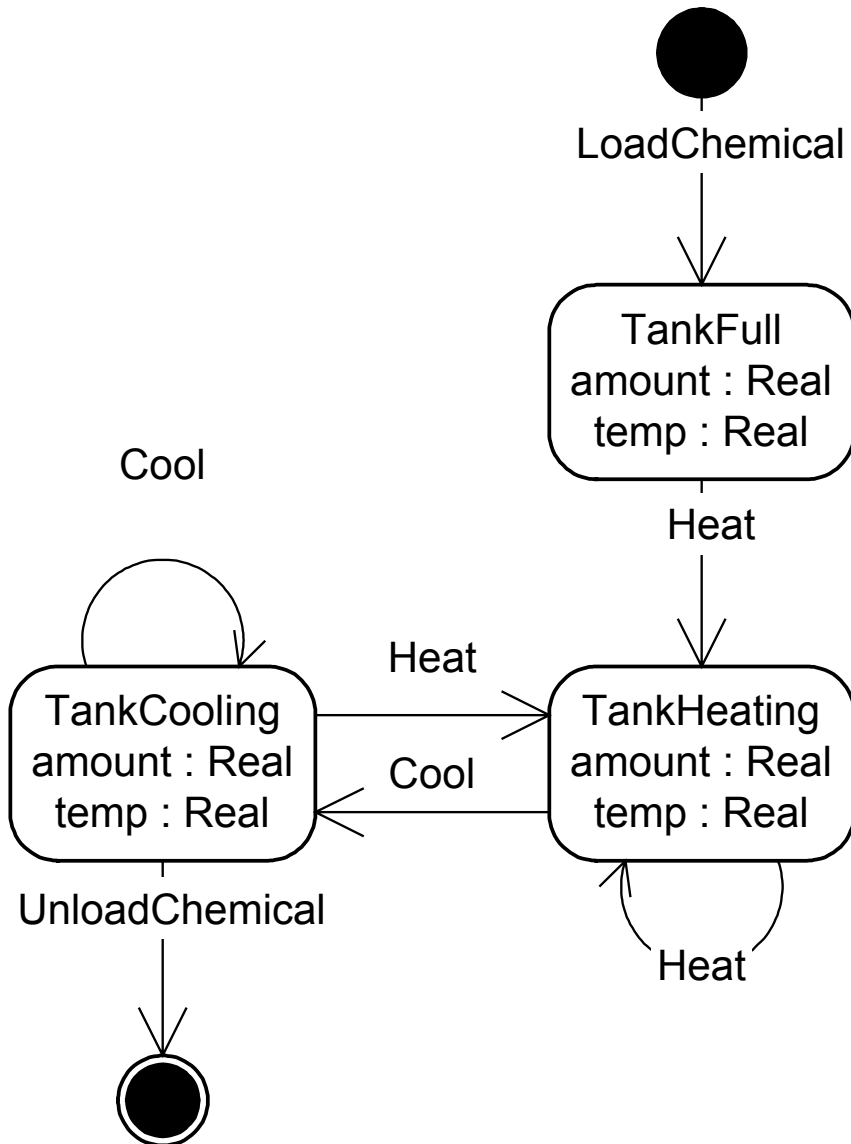
Mapping Theories



Theories can be used to express relationships.

Freely construct a relation and then combine with theories over aspects of the domains.

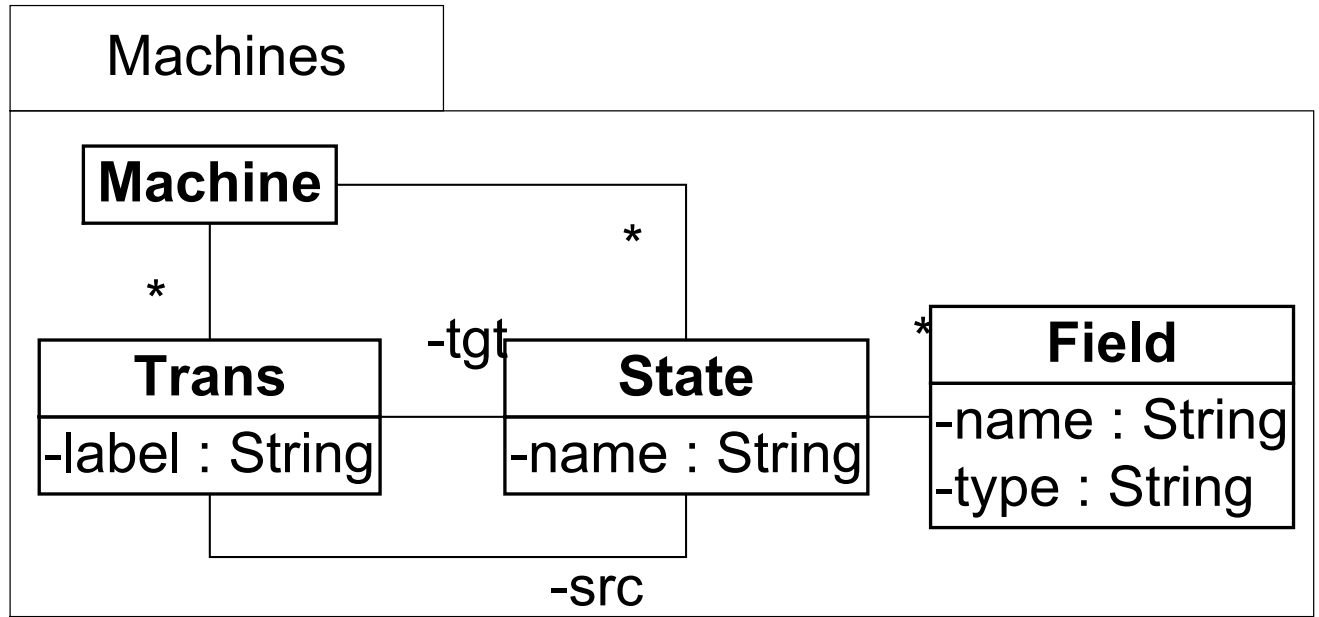
Case Study



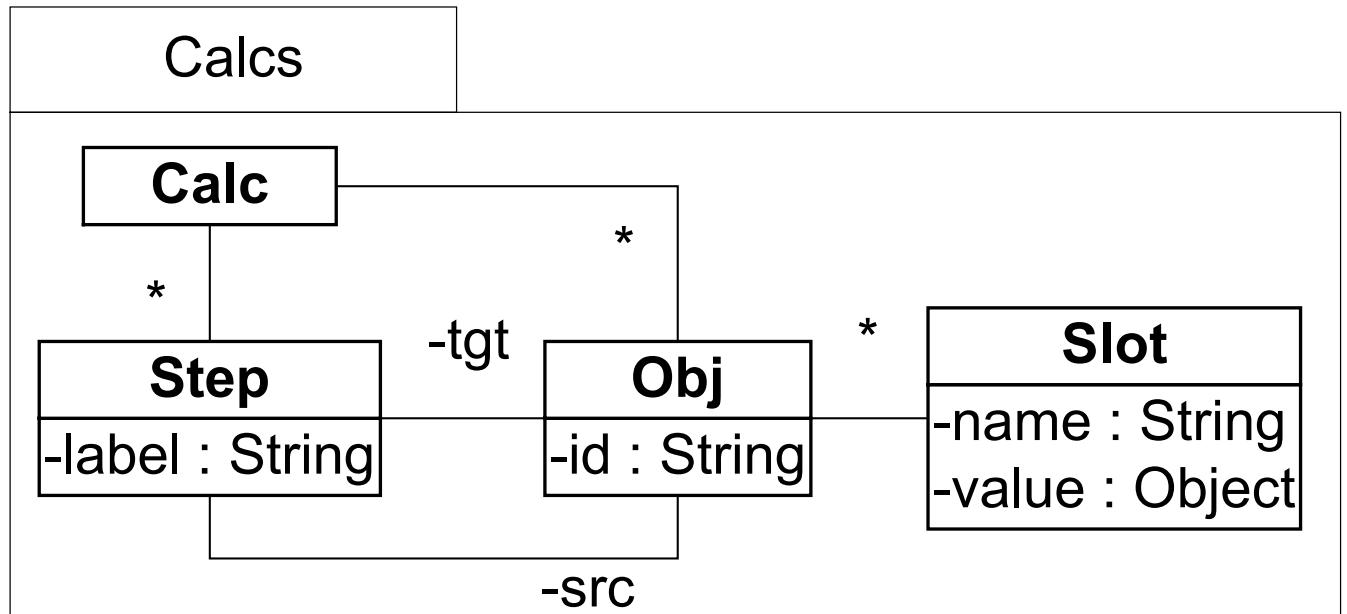
Definition of a simple language for dynamic behaviour.

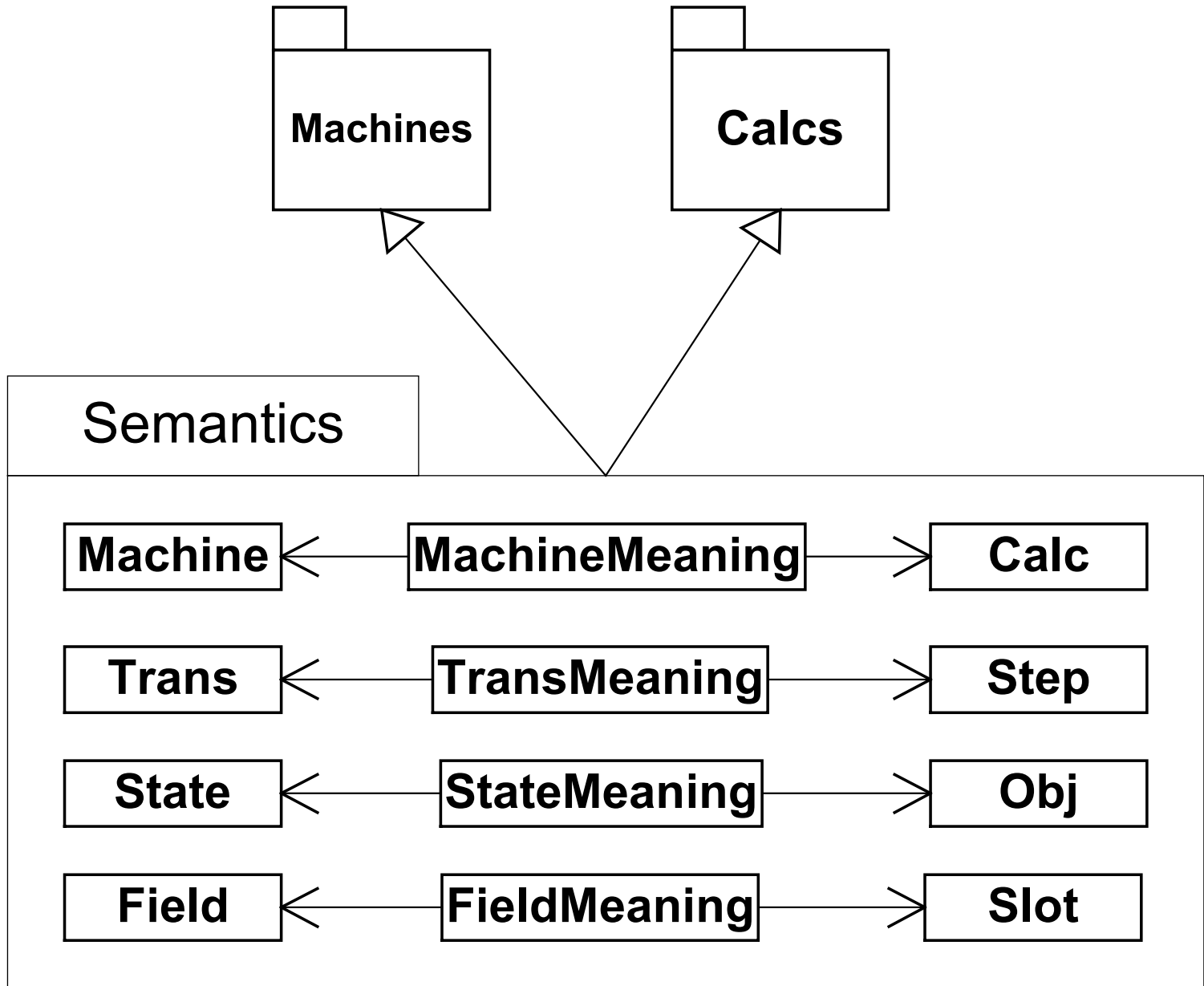
Definition of language given by a meta-model for a language unit.

Syntax
Domain

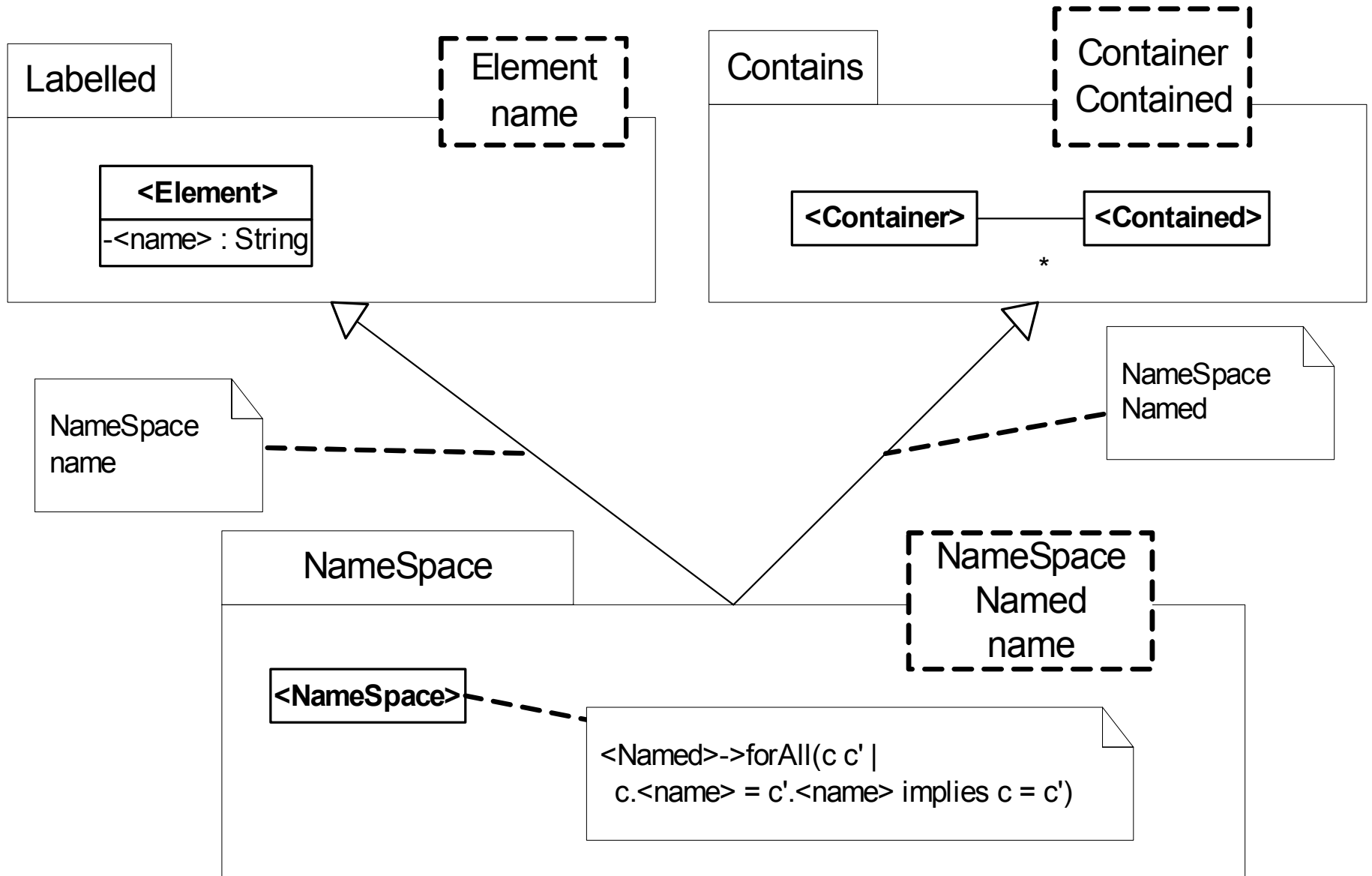


Semantic
Domain

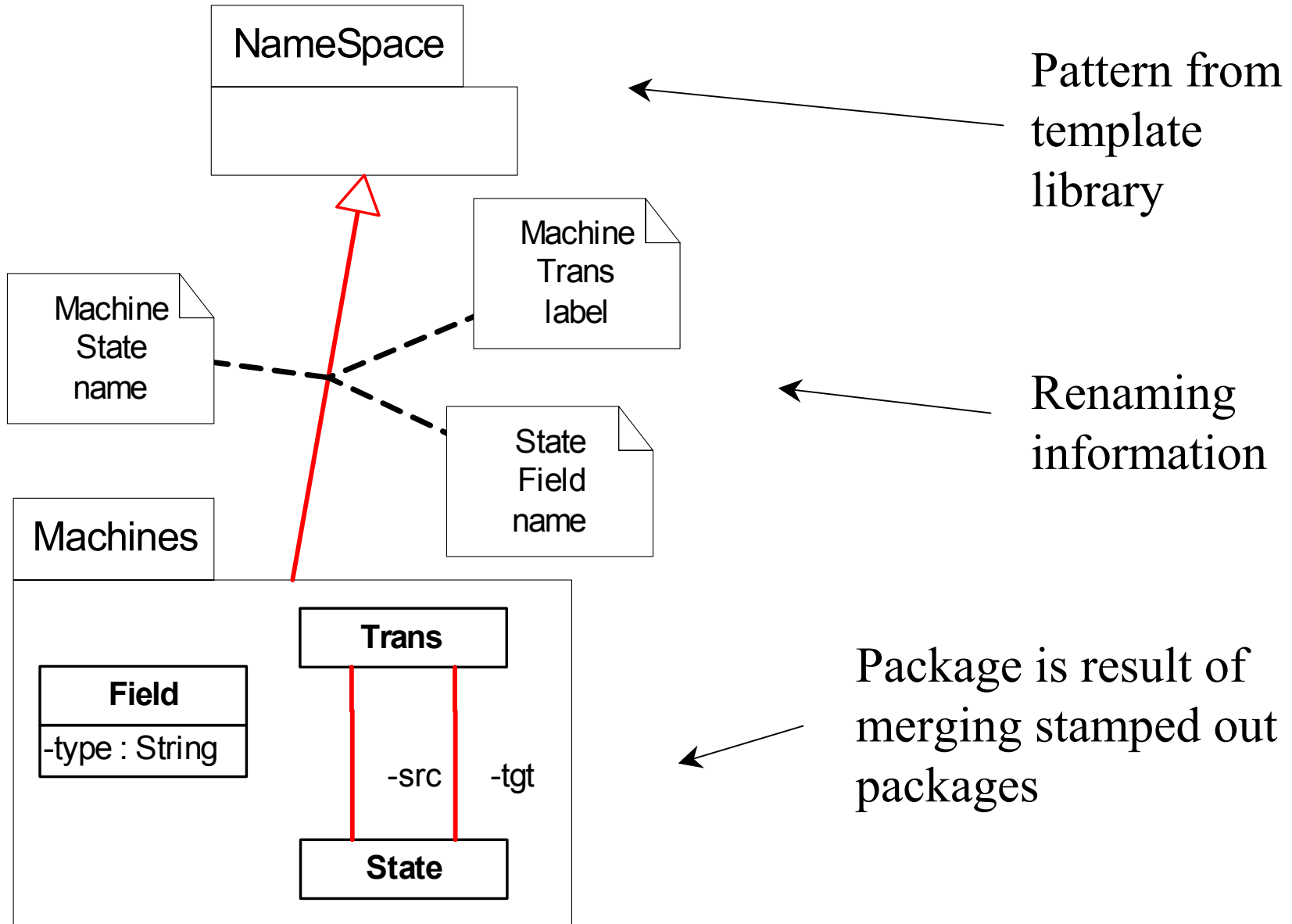




Identify general properties. Express them as templates:



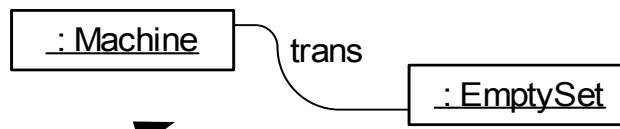
Stamp Out Language Definition



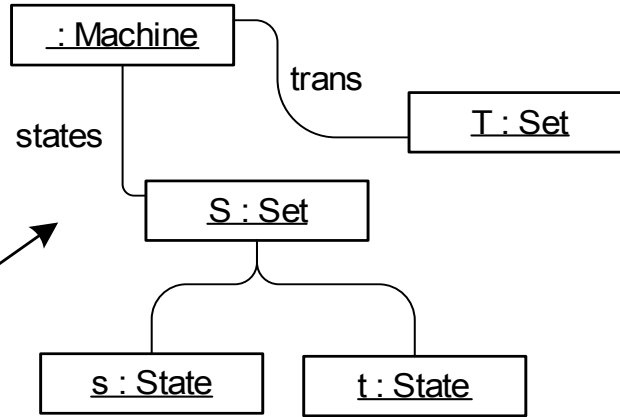
Capture System Properties as Theories

- Machine Language is currently ‘free’.
- Requires constraints on syntax and semantics.
- Example: the source and target of every transition must be a machine state.

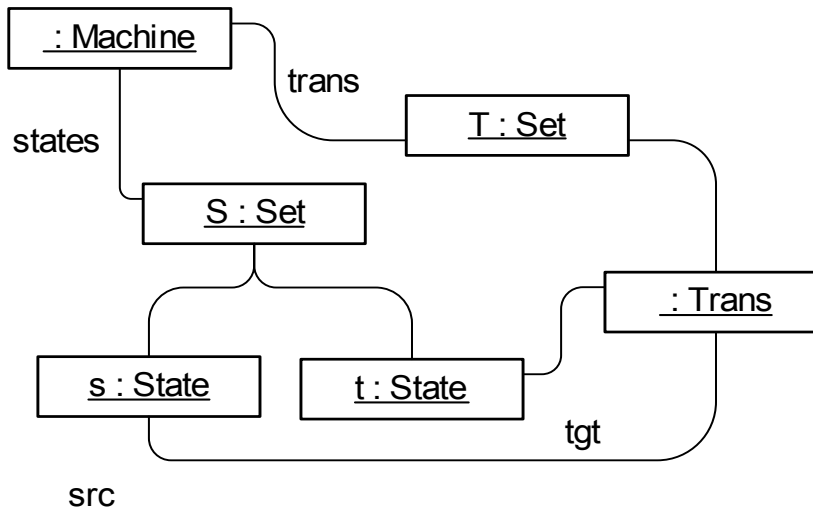
this is OK



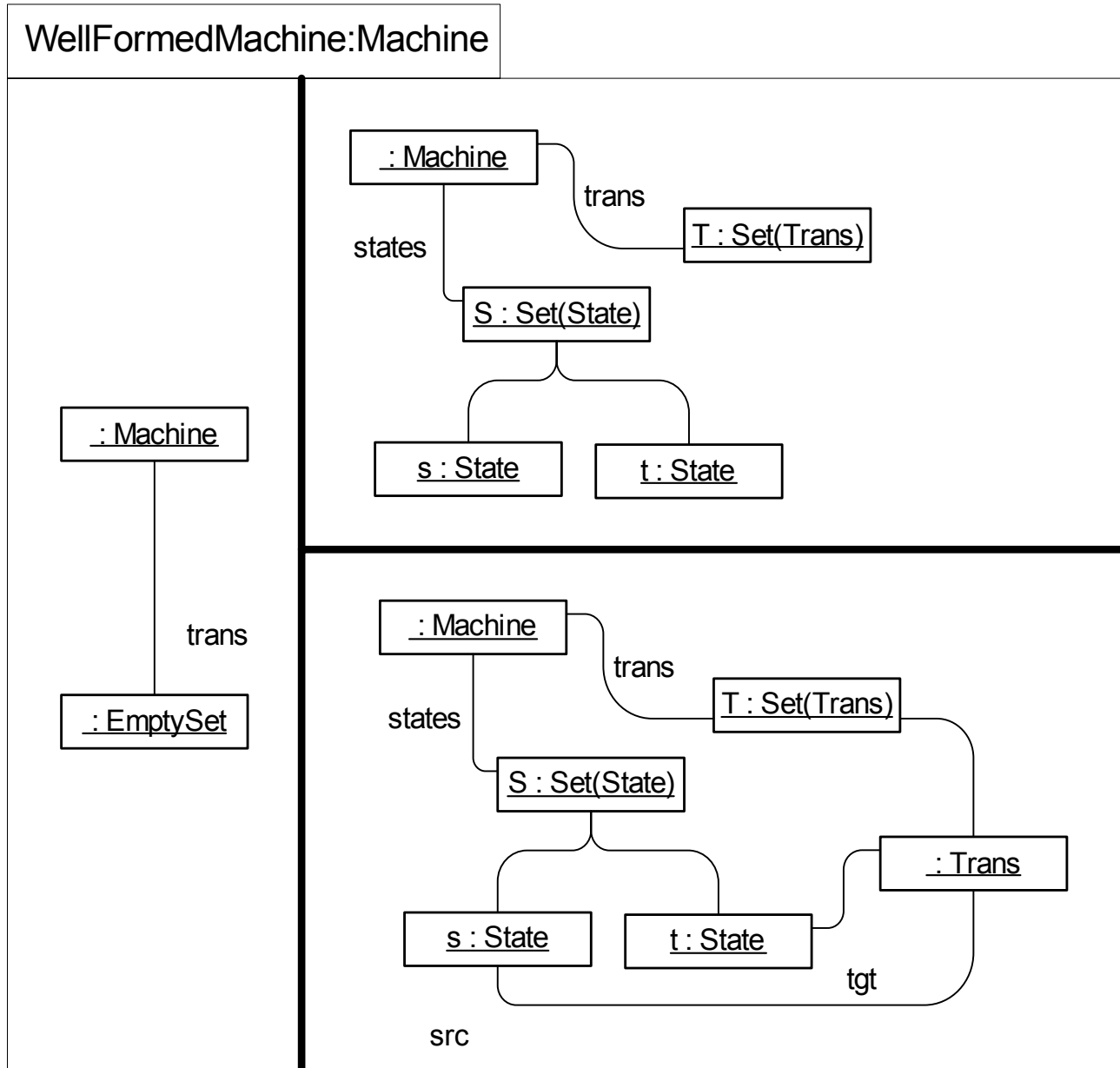
If this is OK



then this is OK



An Object Theory Diagram



Theories are a proposed extension of MML:

theory WellFormedMachine:Machine

or

obj :Machine trans = Set{} end

implies

obj :Machine

trans = T;

states = S->including(s)->including(t)

end

obj :Machine

trans = T->including(obj :Trans src = s; tgt = t end);

states = S->including(s)->including(t)

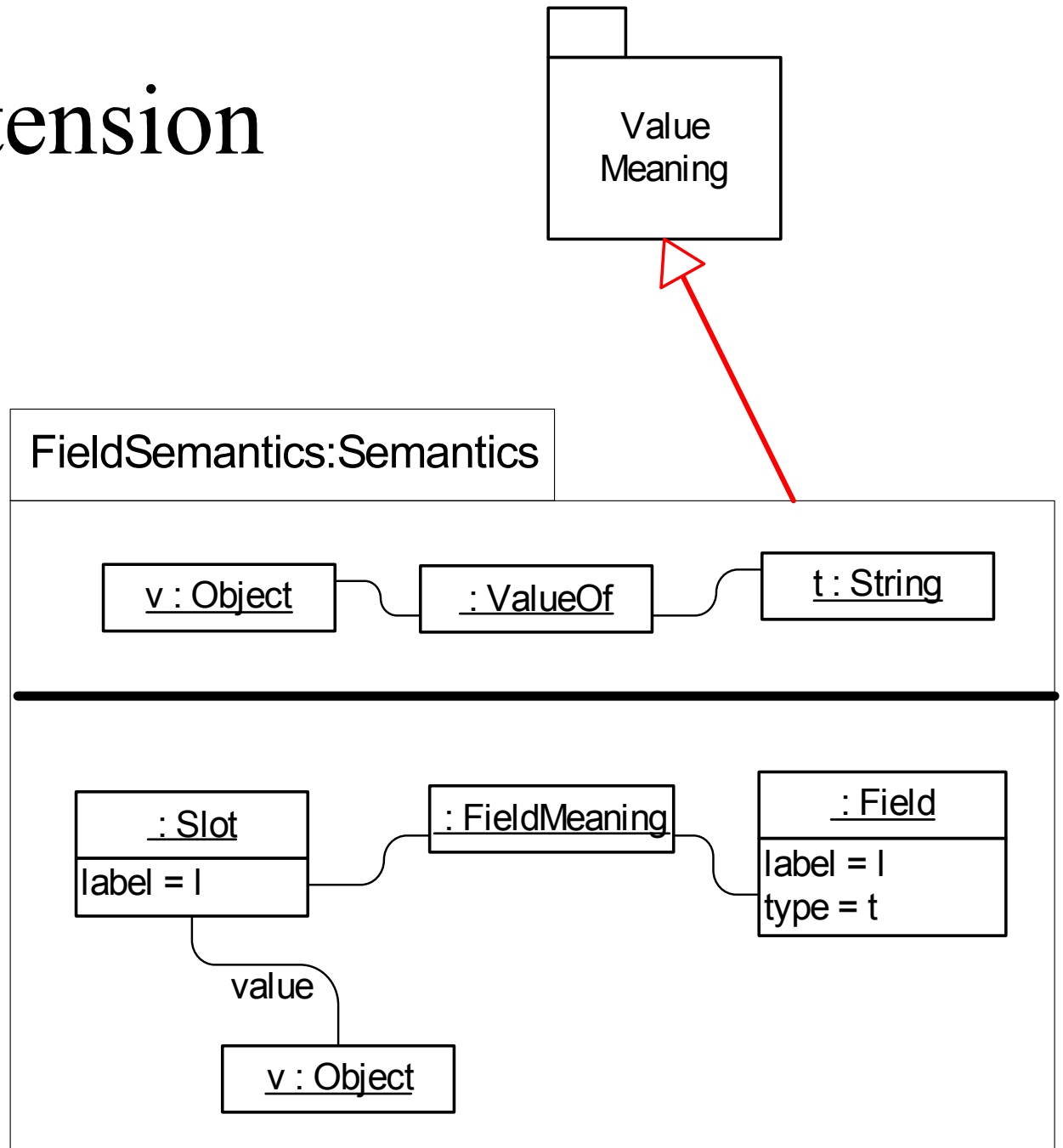
end

end

end

end

Theory Extension



Theories can be defined as extensions of existing theories.

Theories can be used to define relationships between model elements.

Assume a theory **ValueMeaning** that defines **ValueOf** between values and types:

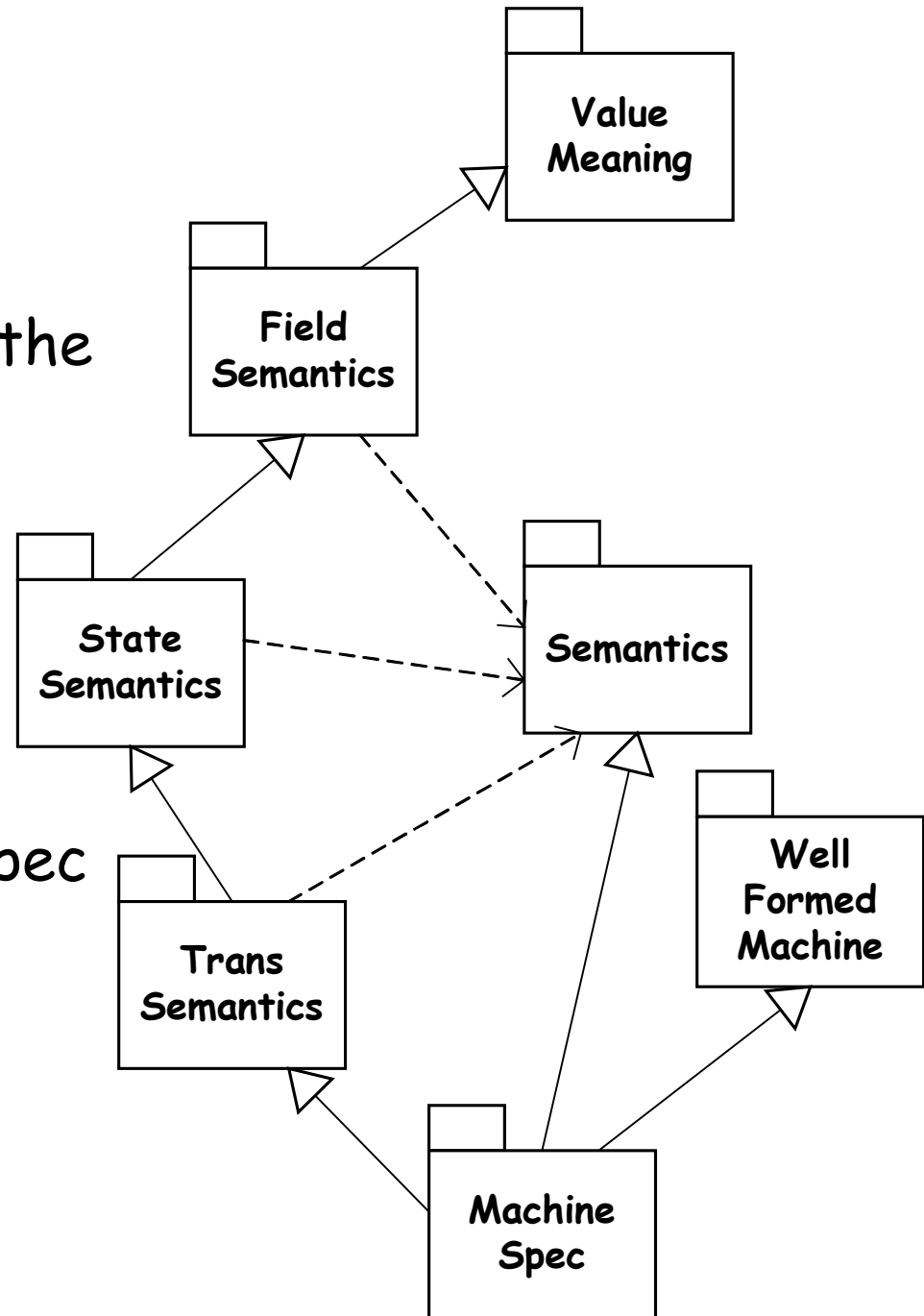
```
theory FieldSemantics:Semantics
  extends ValueMeaning
  implies
    obj :ValueOf
      value = v;
      type = t
    end
    obj :FieldMeaning
      slot = obj :Slot label = l; value = v end;
      field = obj :Field label = l; type = t end
    end
  end
end
```

Define the following theories:

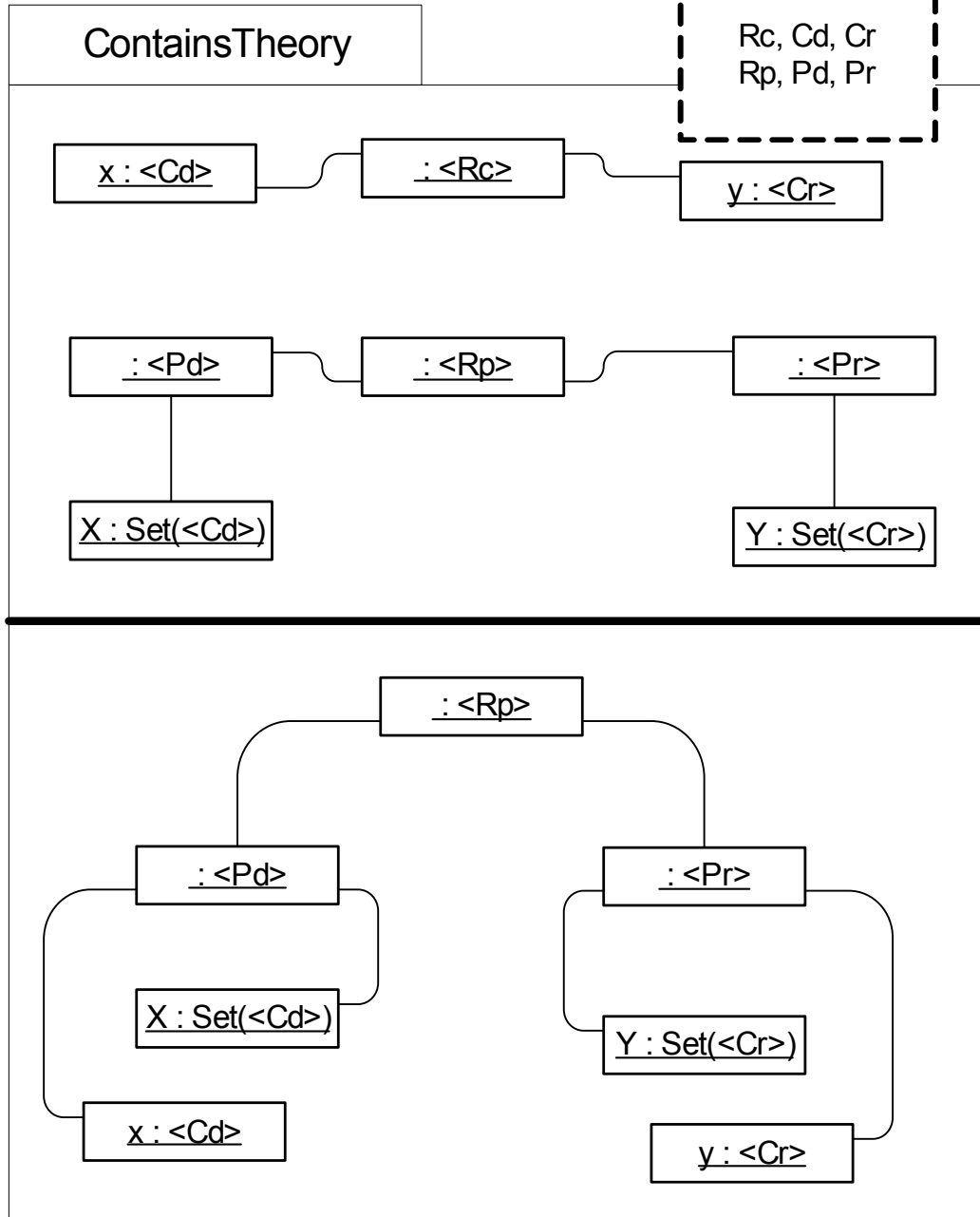
All theories conform to the Semantics.

Theories can specialise existing theories.

The complete MachineSpec is constructed as a combination of the theories over the semantics.



Theory Templates



A relationship R_p holds between two container types P_d and P_r if a relationship R_c holds between all of the contained elements (of types C_d and C_r respectively).

Example: The mapping from a machine to a Java program is composed of sub-mappings from states to Java variables and transitions to Java methods.

Refinement and Refactoring

- Theories and theory templates can be used to capture standard intra and inter language relations.
- Libraries of verified relations.
- Example: containership is refined to Java Vectors.
- Example: inheritance is refactored to become delegation.

Conclusion

- Language Engineering is key for MDA.
- Reusing language aspects.
- Abstract patterns of language properties.
- Relationships and mappings.
- Theories capture reusable, executable properties.