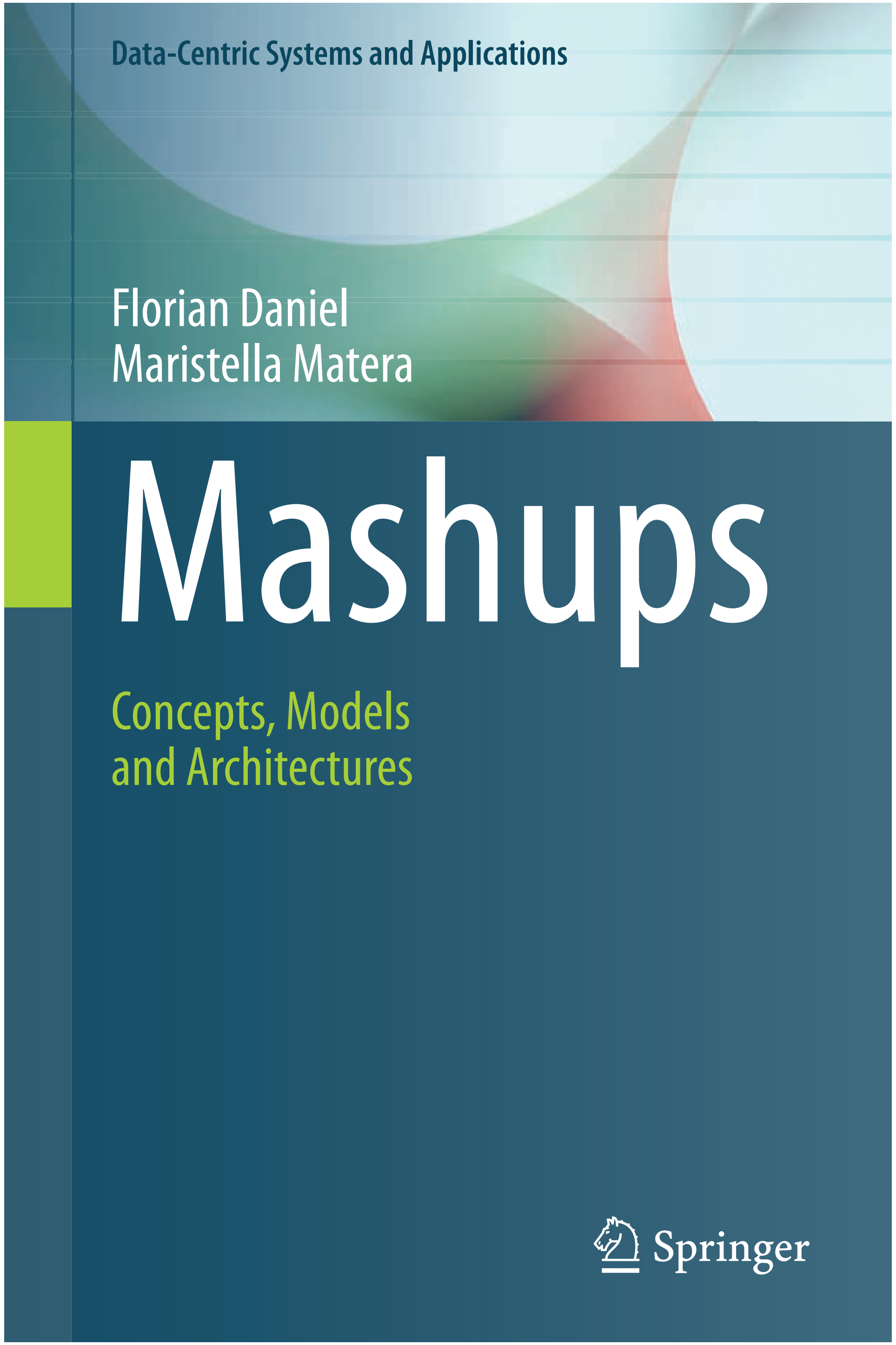

Data-Centric Systems and Applications

Florian Daniel
Maristella Matera

Mashups

Concepts, Models
and Architectures

Springer

Chapter 4
Model-Driven Software Development

Figures

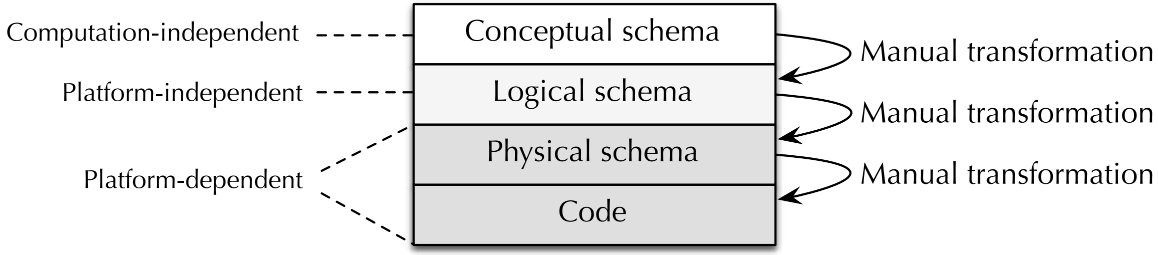

Fig. 4.1 The different schemas in the conceptual modeling stack for database design.

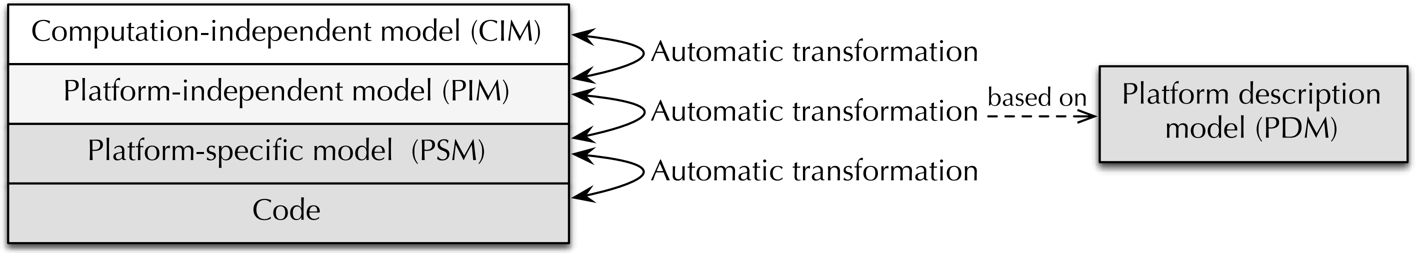
Computation-independent model (CIM)
Platform-independent model (PIM)
Platform-specific model (PSM)
Code
Automatic transformation
Automatic transformation
Automatic transformation
based on
Platform description model (PDM)

Fig. 4.2 Models and transformations in the Model-Driven Architecture (MDA) [195].

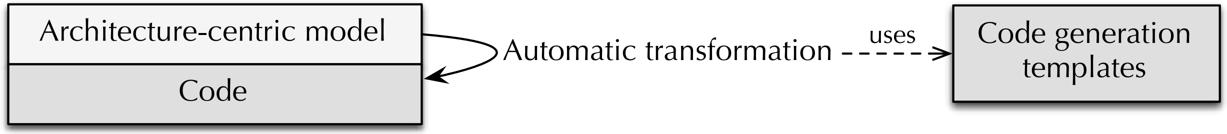

Fig. 4.3 The ingredients of architecture-centric model-driven software development (AC-MDSD) according to Stahl and Völter [256].

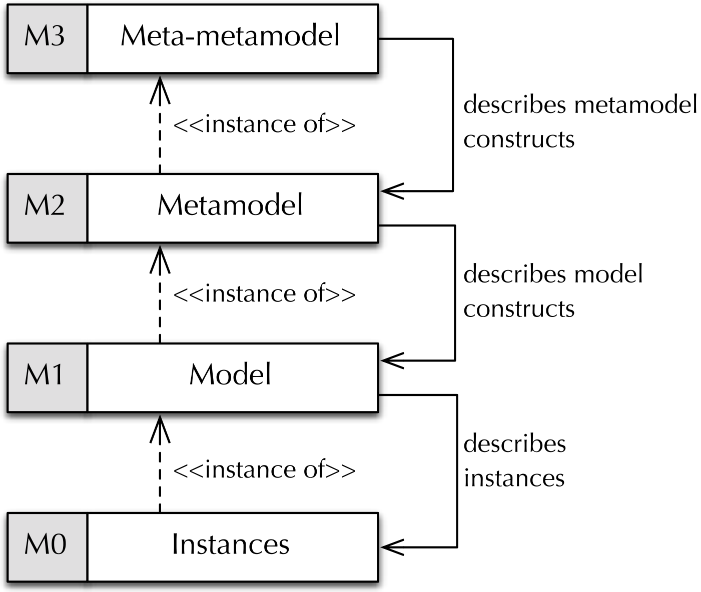

Fig. 4.4 The four metalevels proposed in OMG's Meta Object Facility [215].

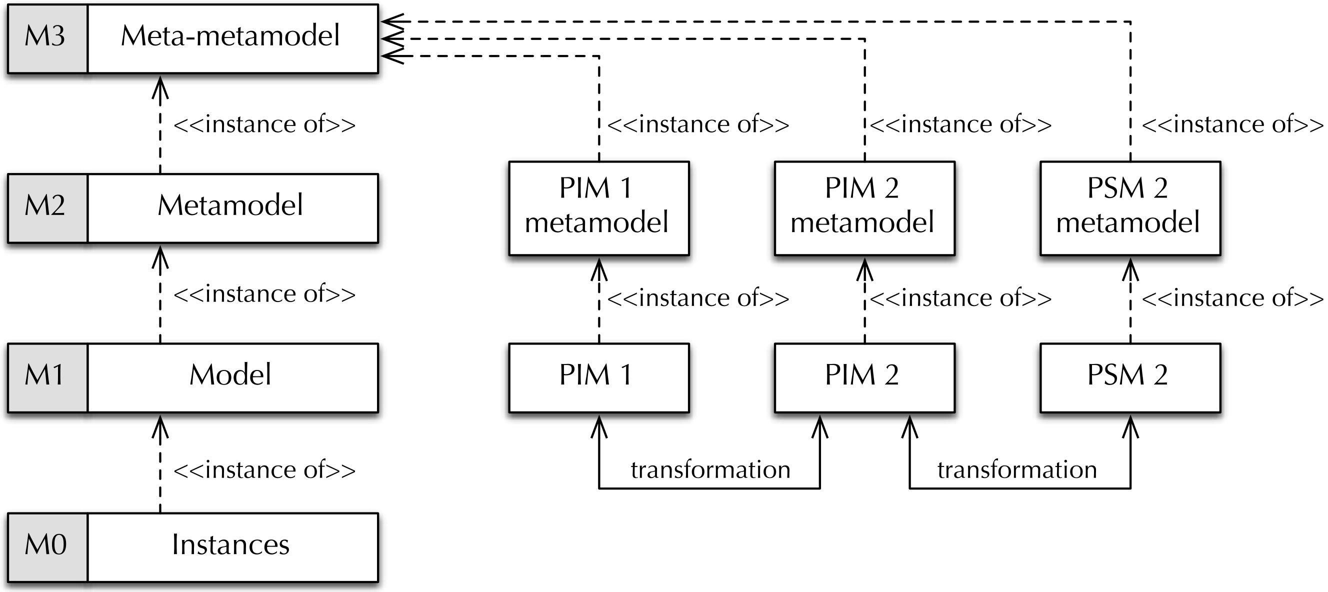

Fig. 4.5 Meta levels vs. abstraction levels. Abstraction levels express different abstractions of a same artifact (the application) at metalevel M1; metalevels express different artifacts (instances, model constructs, metamodel constructs).

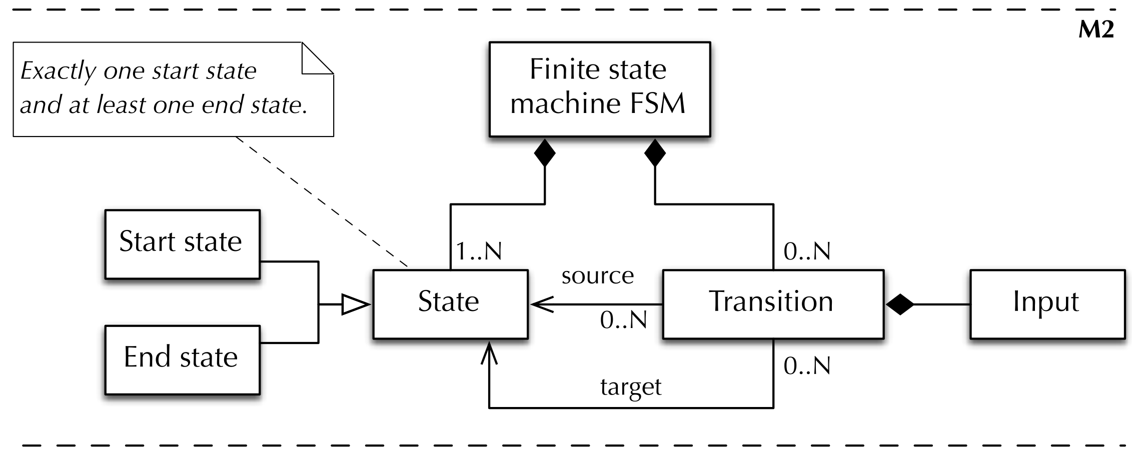

Fig. 4.6 A platform-independent M2 metamodel for a finite state machine with start and end states. The FSM triggers its transitions upon the reception of an input character.

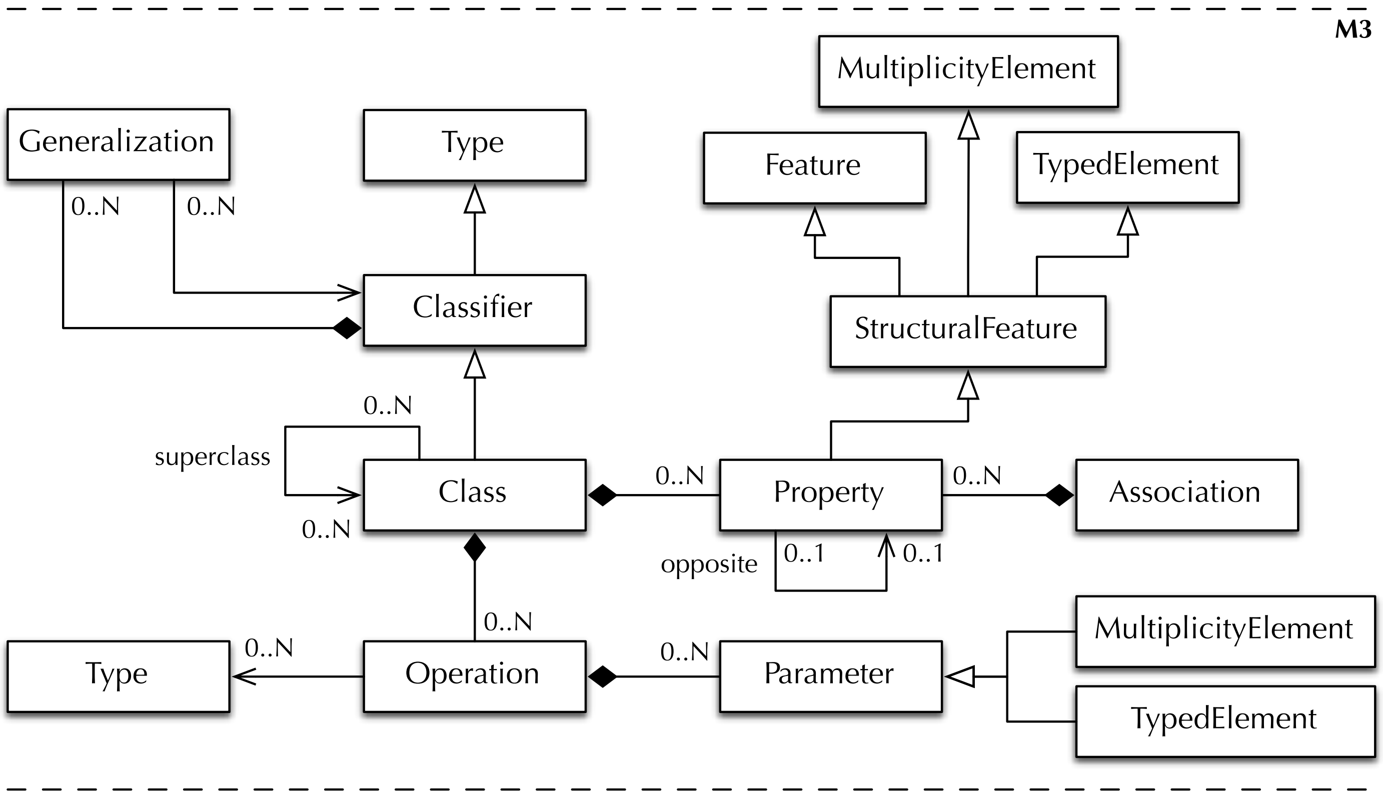

Fig. 4.7 A simplified version of the EMOF package, the core of the MOF [215].

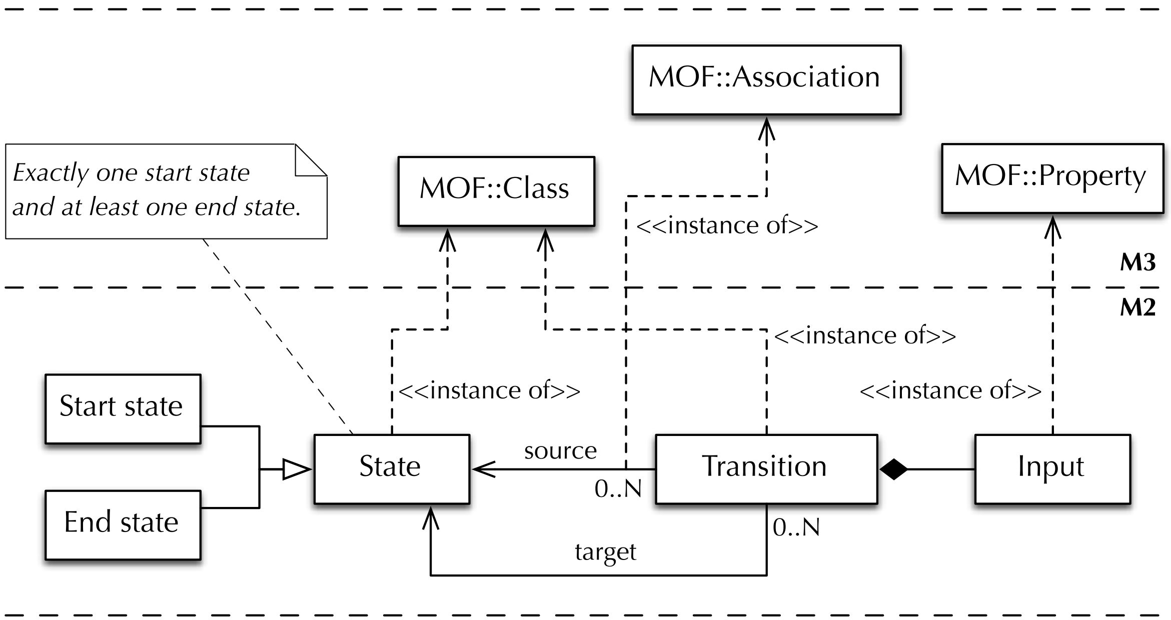

Fig. 4.8 A metamodel for a FSM defined as instance of the Meta Object Facility (MOF) [215] with some `<<instance of>>` relationships highlighted: states and transitions are instances of `MOF::Class`, relationships of `MOF::Association`, and attributes of `MOF::Property`.

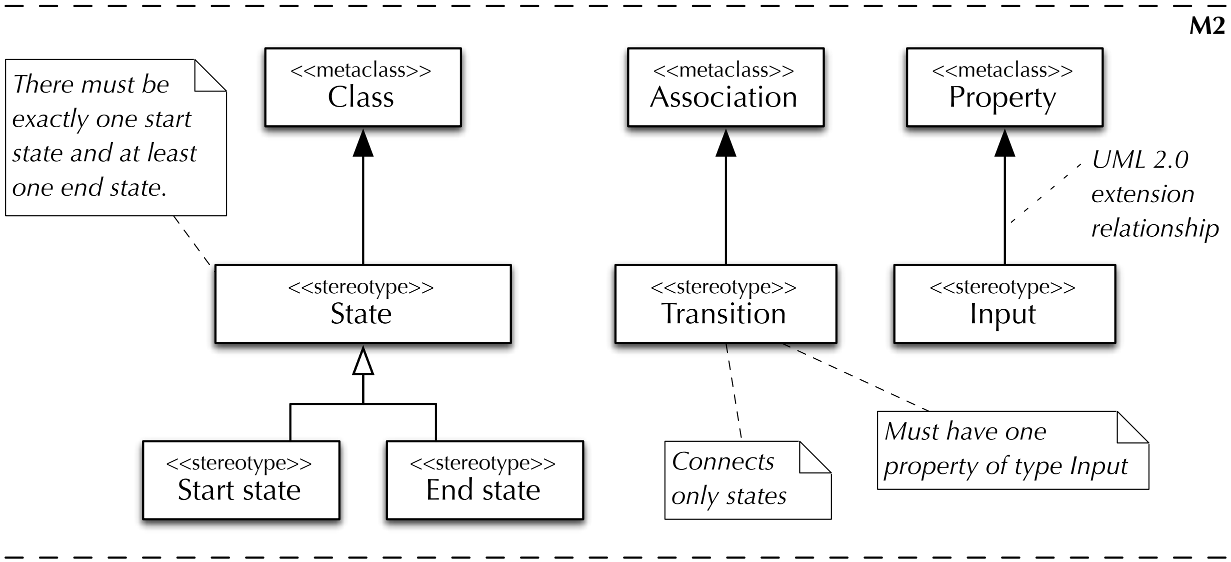
M2
There must be exactly one start state and at least one end state.
<<metaclass>> Class
<<metaclass>> Association
<<metaclass>> Property
UML 2.0 extension relationship
<<stereotype>> State
<<stereotype>> Transition
<<stereotype>> Input
<<stereotype>> Start state
<<stereotype>> End state
Connects only states
Must have one property of type Input

Fig. 4.9 A metamodel for a FSM defined as UML 2.0 profile. Syntactically, UML profiles are at the metalevel M1, yet semantically they are at metalevel M2, as profiles specialize the UML metamodel (e.g., `UML::Class`).

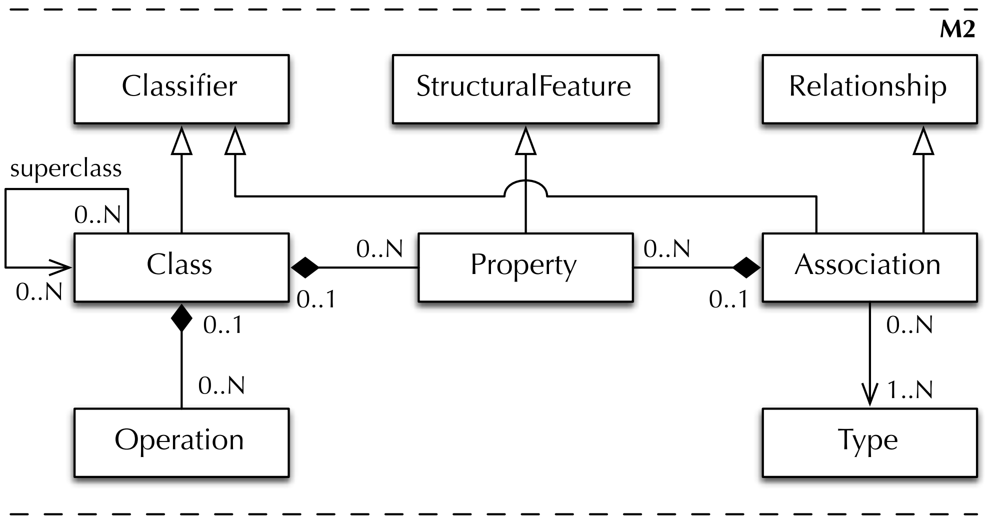

Fig. 4.10 A simplified excerpt of the Classes diagram of the Constructs package of the UML metamodel [217], the starting point for the definition of UML profiles.

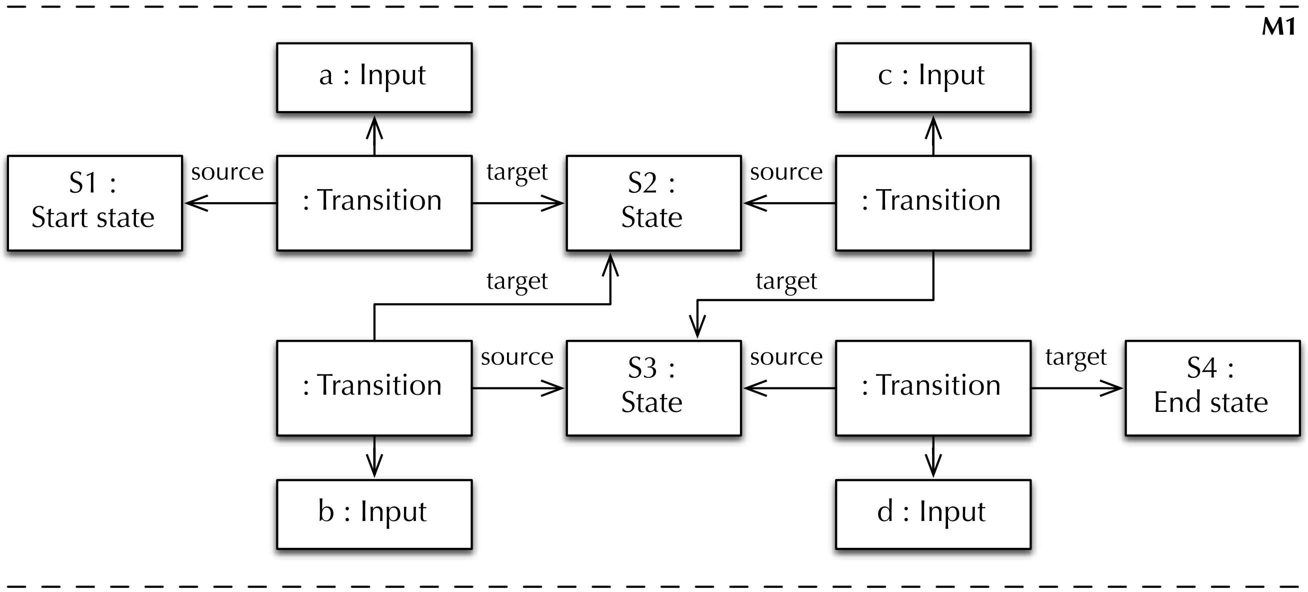

Fig. 4.11 A model instance of the finite state machine defined in Figure 4.6 using a UML object diagram as modeling syntax.

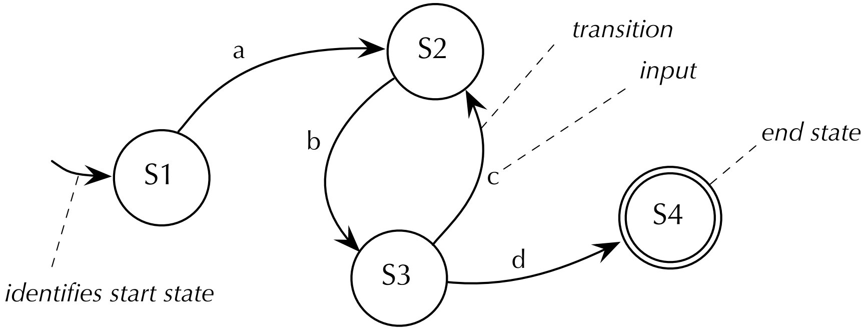

Fig. 4.12 A model instance of the finite state machine defined in Figure 4.6 using an own, more intuitive modeling syntax.

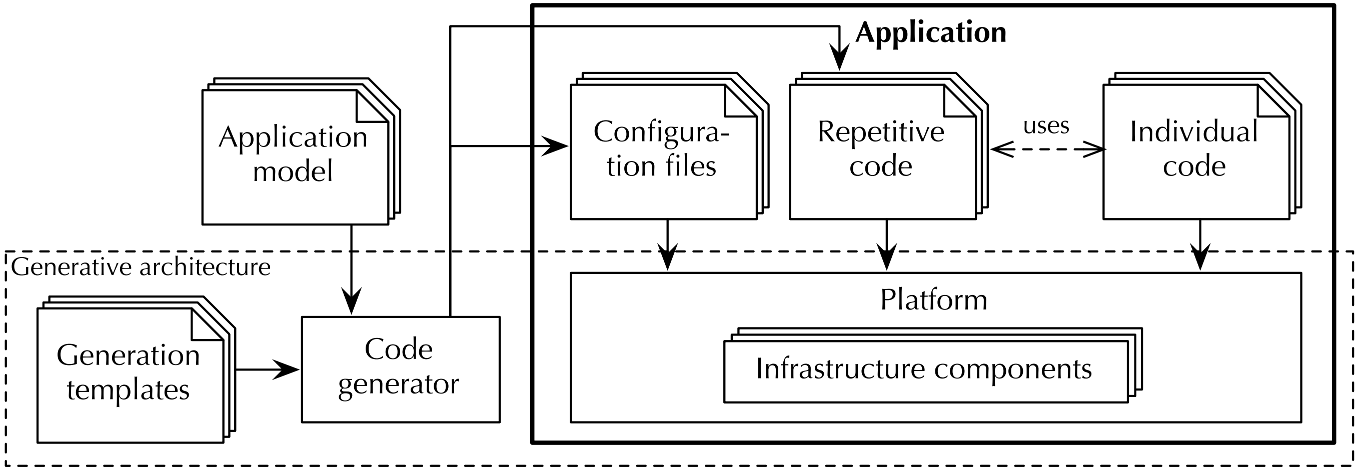

Fig. 4.13 The typical MDSD code generation process based on code templates. Depending on the platform features and the modeled application, either configuration files or code or both are generated; individual code may be plugged in manually.

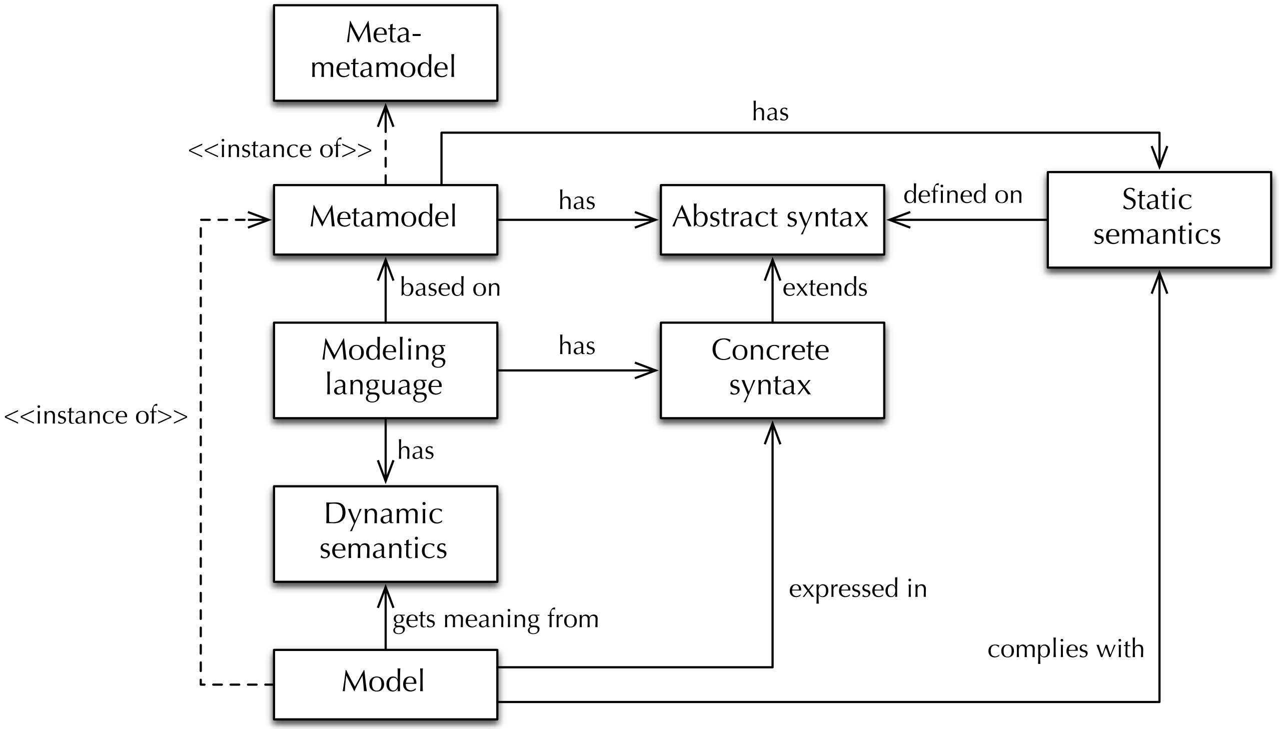

Fig. 4.14 Summary of the key concepts of MDSD.