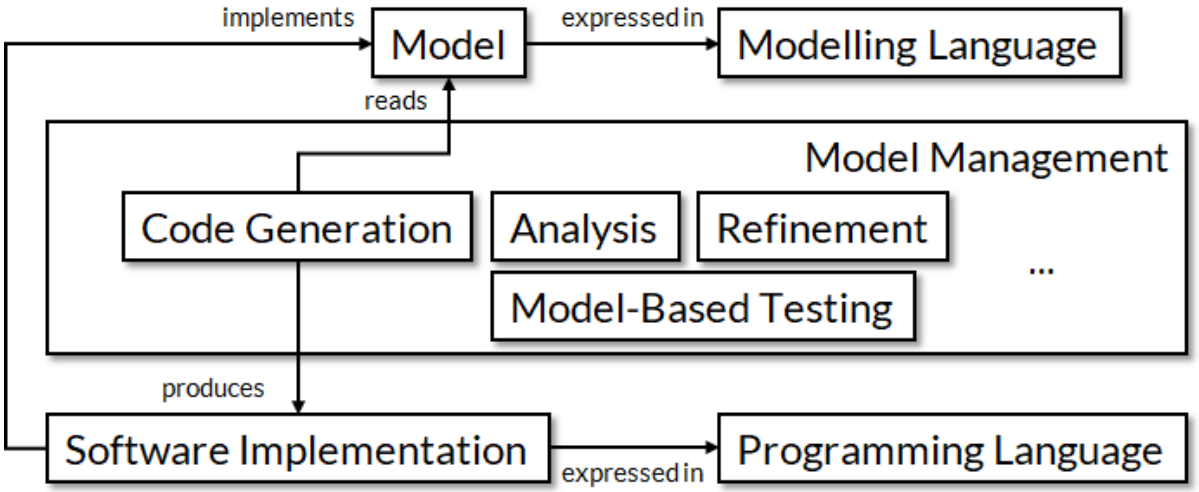
A glossary of model-driven engineering

Model-driven engineering (MDE), the thing that MDENet is all about, has been known under several different names and acronyms over the years, and you may find yourself confused trying to make sense of the plethora. We are here to help with this glossary of common alternative labels for MDE. If you have heard about any of the below, chances are you will find something of interest in the MDENet community.

To help with the explanations, the figure below gives a summary of some key terms in MDE. MDE focuses on the development of software via so-called "models", high-level representations of an aspect of the software solution. Models are expressed in so-called "modelling languages"; these can be text-based languages or can be graphical, diagrammatic languages. Often, modelling languages will be developed for a specific purpose in a specific domain (so-called domain-specific modelling languages or DSMLs), but general-purpose modelling languages also exist. "Software implementations" (also known as programs) in a "programming language" are then used to "implement" the solution captured by a model or a set of models. This implementation may be created manually, but sometimes is created through an automated process called "code generation". Code generation is one type of model-management activity. Other model-management activities include analysis, refinement or model-based testing. Different variants of MDE differ in how they interpret and realise these concepts. We list the most prominent different variants below and try to explain their differences; though it is worth noting that some of these labels have been used interchangeably in the past.



1. *Model-Driven Engineering (MDE).* This label most typically describes the general idea of developing high-level modelling languages (often, but not always, domain-specific modelling languages) that are then used to describe different aspects of a

software(-intensive) system in models, undertake predictive analysis and validation on these models (by using semi-automated analysis and reasoning tools), communicate and change design decisions, and, finally, generate a software implementation.

2. *Model-Driven Development (MDD).* This is sometimes used as a synonym for MDE.
3. *Model-Based Engineering/Development (MBE/MBD).* This is a subset of MDD/MDE where there is less automation in the production of the final software implementation. Software is still designed through models primarily, the production of the final software implementation remains manual rather than using automated code generation.

   Computer Aided Software Engineering (CASE) was a precursor of the MBE/MBD form of software development, peaking in the 1990s. MBE/MBD approaches are often based on general-purpose modelling languages; that is modelling languages that reflect general coding concepts rather than domain-specific concepts about the problem to be solved. The *Unified Modelling Language (UML)* is the most well-known such "general purpose" modelling language, which emerged from the "methods war" of the early 1990s and has since grown as an OMG standard. Early MDE work focused primarily on generating code from UML (or making UML executable via interpretation). Later work looked at allowing UML to be customised for specific domains through profiles and stereotypes. Most recent work in MDE appears to focus primarily on domain-specific modelling languages.

4. *Model-Based Systems/Software Engineering (MBSE).* Initially, the S stood for "systems" and emphasised the fact that many software-intensive systems are more than just software and that there is a need to integrate the modelling of the software with the modelling of the remainder of the wider system. SysML is a prominent modelling language in this particular form of MDE. More recently, the abbreviation has also seen use as a shortening of the "software" variant. Similar to MBSE, there are also MDSE--where the S is sometimes for Software and sometimes for Systems--and MDSD.
5. *Model-Driven Architecture (MDA).* This is a very specific form of model-driven engineering focusing primarily on the incremental refinement of models from "*computation-independent models*" (CIMs, models at the requirements level) via "*platform-independent models*" (PIMs) to "*platform-specific models*" (PSMs) from which the final source code can be generated. MDA is an OMG standard that was a key driver for technological innovations like the MOF 4-layer architecture and for a good deal of initial interest in MDE. Its central promise was to address vendor lock-in by allowing for the generation of different PSMs from the same PIM, thus reducing the cost of changing platform vendor to the cost of producing (or procuring) a different model-to-model transformation and code generator.
6. *Lowcode.* Lowcode has a largely parallel history, tracing back to innovations like Java Beans or Borland's Delphi. Some modern-day vendors of lowcode technology have a direct MDE background, however. A key promise of lowcode is to allow visual development of applications by plugging together existing components. The technology has strong foundations in component-based software engineering and typically defines a standard configuration interface / API for the components enabling the lowcode platform to easily ingest new components and make them available to application developers. Lowcode platforms (e.g., Microsoft Power Apps, Google AppSheet or

Outsystems) typically provide support both for designing and implementing a piece of software and hosting it so it can be made available to end users via the Internet. Lowcode often targets “end-user programming” or “citizen developers”.