Domain Engineering is an engineering discipline concerned with building reusable assets, such as specification sets, patterns, and components, in specific domains. A domain in this context can be defined as an area of knowledge that uses common concepts for describing phenomena, requirements, problems, capabilities, and solutions. The purpose of domain engineering is to identify, model, construct, catalog, and disseminate artifacts that represent the commonalities and differences within a domain. Although being applicable to different engineering disciplines, domain engineering methods and domain specific languages (DSL) receive nowadays special attention from the information systems and software engineering researchers and practitioners who deal with artifact reuse, application validation, and domain knowledge representation. In particular, these topics are of interest in the areas of software product line engineering and ontology engineering. One of the reasons for this interest is the increasing variability of information and software systems and the need to obtain and share expertise in different, evolving domains.

Domain engineering deals with two main layers: the domain layer, which deals with the representation of domain elements, and the application layer, which deals with the software applications and information systems artifacts. In other words, the programs, applications, or systems are included in the application layer, whereas their common and variable characteristics, as can be described, for example, by patterns or emerging standards, are generalized and presented in the domain layer.

Similarly to application engineering, domain engineering includes three main activities: domain analysis, domain design, and domain implementation. Domain analysis identifies a domain and captures its ontology. Its aim is to specify the basic concepts of the domain, identify the possible relationships among these concepts, and represent this understanding in a useful way. Domain design and domain implementation are concerned with mechanisms for translating requirements to artifacts that will operate in the domain, i.e., into systems that are made up of components with the intent of reusing them to the highest extent possible. All these
activities are performed within the domain layer. However, domain engineering also supports inter-layer activities, namely interactions that exist between the domain and application layers. Specifically, domain layer artifacts may be reused and used for the design and validation of the specifications of application layer artifacts, while the applications may be generalized into domain artifacts in a process that can be termed knowledge elicitation. Figure 1 visually summarizes the two layer model of domain engineering and its related activities.

Domain engineering as a discipline has practical significance as it can provide methods and techniques that may help reduce time-to-market, product cost, and projects risks on one hand, and help improve product quality and performance on a consistent basis on the other hand.

Figure 1. The two layer model of domain engineering