1.4 WHAT IS DSM?

Domain-Specific Modeling mainly aims to do two things. First, raise the level of abstraction beyond programming by specifying the solution in a language that directly uses concepts and rules from a specific problem domain. Second, generate final products in a chosen programming language or other form from these highlevel specifications. Usually the code generation is further supported by framework code that provides the common atomic implementations for the applications within the domain. The more extensive automation of application development is possible because the modeling language, code generator, and framework code need fit the requirements of a narrow application domain. In other words, they are domainspecific and are fully under the control of their users.

1.5 WHEN TO USE DSM?

Languages and tools that are made to solve the particular task that we are working with always perform better than general-purpose ones. Therefore DSM solutions should be applied whenever it is possible. DSM is not a solution for every development situation though. We need to know what we are doing before we can automate it. A DSM solution is therefore implausible when building an application or a feature unlike anything developed earlier. It is something unique that we don’t know about. In such a situation we usually can only make prototypes and mock-up applications and follow the trial-and-error method, hopefully in small, agile, and iterative steps. In reality, we don’t often face such unique development situations. It is much more likely that after coding some features we start to find similarities in the code and patterns that seem to repeat. In such situations, developers usually agree that it does not make sense to write all code character by character. For most developers, it would then make sense to focus on just the unique functionality, the differences between the various features and products, rather than wasting time and effort reimplementing similar functionality again and again. Avoiding reinventing the wheel is good advice for a single developer, but even more so if colleagues are implementing almost identical code too. In code-driven development, patterns can evolve into libraries, reusable components, and services to be used. Building a DSM solution requires a similar mindset as it offers a way to find a balance between writing the code manually and generating it. How the actual decision is made differs between application domains. In Part III, we describe five DSM cases where the partitioning is done in various ways, and in Part IV, we give guidelines on how you can do it. 18 INTRODUCTION Using resources to build a DSM solution implies that development work is conducted over a longer period within the same domain. DSM is therefore a less likely option for companies that are working in short term projects without knowing which kind of application domain the next customer has. Similarly, it is less suitable for generalist consultancy companies and for those having their core competence in a particular programming language rather than a problem domain. Although the time to implement a DSM solution can be short, from a few weeks to months, the expected time to benefit from it can decrease the investment interest. The longer a company can predict to be working in the same domain, the more likely it will be interested in developing a DSM solution. Some typical cases for DSM are companies having a product line, making similar kinds of products, or building applications on top of a common library or platform. For product lines, a typical case of using domain-specific languages (e.g., Weiss and Lai, 1999) is to focus on specifying just variation: how products are different. The commonalities are then provided by the underlying framework. For companies making applications on top of a platform, DSM works well as it allows having languages that hide the details of the libraries and APIs by raising the level of abstraction on which the applications are built. Application developers can model the applications using these high level concepts and generate working code that takes the best advantage of the platform and its services. DSM is also suitable for situations where domain experts, who often can be nonprogrammers, can make complete specifications using their own terminology and run generators to produce the application code. This capability to support domain experts’ concepts makes DSM applicable for end-user programming too