### APPENDIX A

## METAMODELING LANGUAGE

A metamodeling language is used to specify the abstract syntax of modeling languages. In this appendix we describe the main concepts of the metamodeling language used in this book. For some background on describing modeling languages, see Section 4.2.1. To understand the basic principles of metamodeling, see Section 4.2.4. More details and some more advanced topics on metamodeling can be found in Sections 10.3–10.5.

Throughout the book, we have consciously avoided limiting the discussion to a particular metamodeling language or tool, but when discussing concrete cases or details of a particular technique it is important to have a precise, consistent set of concepts. In the example cases in Part III, and occasionally in the discussions in Part IV, we therefore apply a widely used metamodeling language, GOPRR.

Although more familiar languages from other domains can also capture some of the information necessary in metamodels, GOPRR is preferred here because it was specifically designed for describing modeling languages. Using a domain-specific modeling language for the task of metamodeling brings the same benefits as the use of DSM in any domain: simplicity, precision, and automation. Languages like UML, MOF, and ER were intended for other domains, entailing greater effort, loss of precision, and risk of misunderstanding for the reader. These issues are discussed further in Section 14.3.1, which looks at what is needed in a metamodeling language.

Before looking at metamodels, we will start with an example model of a simple order handling system (Fig. A.1).

Domain-Specific Modeling: Enabling Full Code Generation, Steven Kelly and Juha-Pekka Tolvanen Copyright # 2008 John Wiley & Sons, Inc.

411

METAMODELING LANGUAGE

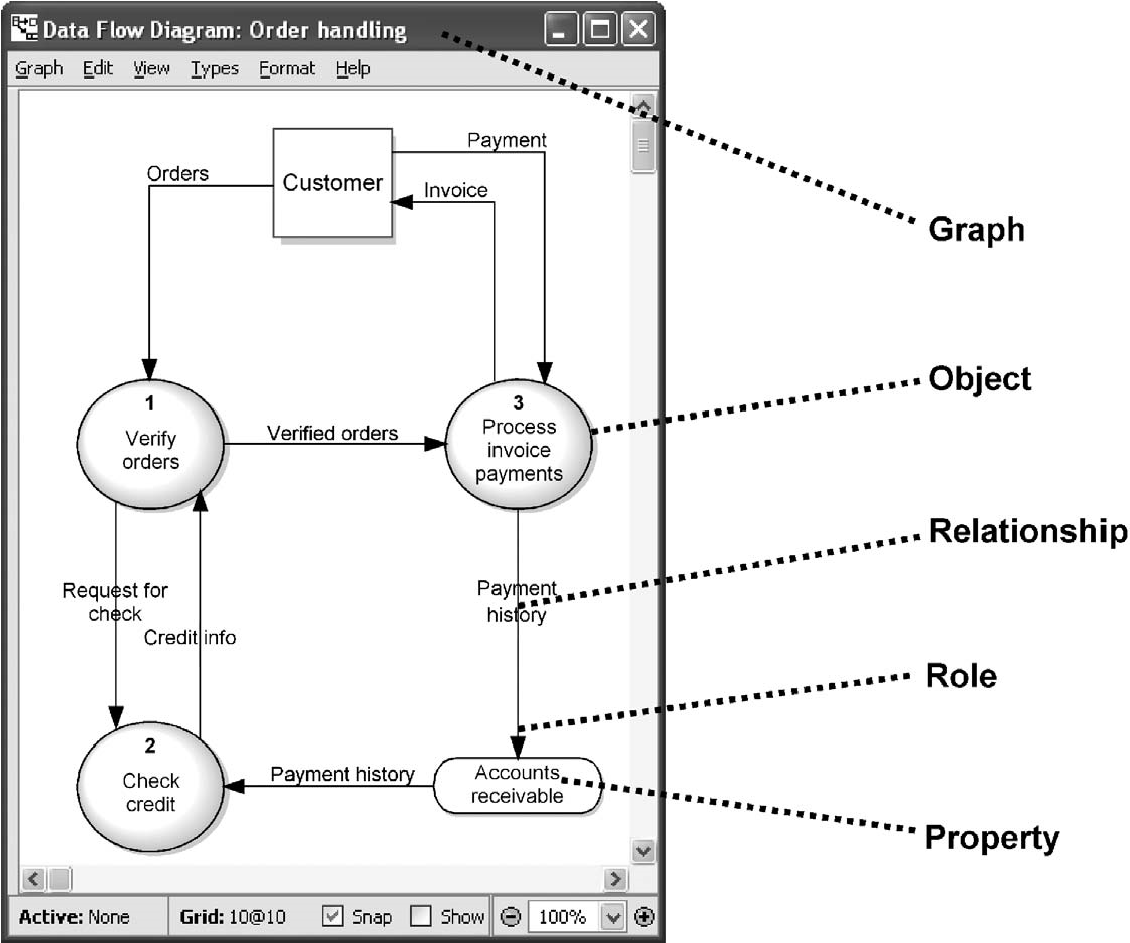


FIGURE A.1 A data flow diagram of order handling system (partial)

This diagram describes the flow of data when handling orders and invoices. On the right of the figure, we have labeled some examples of the basic constructs that can be found in all models, whatever their language.

. A graph is one individual model, often shown as a diagram

. Objects are the main elements of graphs, often shown as boxes or circles . A relationship connects objects together, often shown as a label over the connection . A role connects an object into a relationship, shown as a line and often an arrow-head

. A property is an attribute characterizing one of the above, often shown as a label

TABLE A.1 Instances, Types, and Metatypes

|  |  |  |
| --- | --- | --- |
| Instance | Type | Metatype |
| Order handling | Data flow diagram | Graph |
| Process invoice payments, customer | Process, external | Object |
| Payment history, verified orders | Data flow | Relationship |
| Arrow-head at accounts receivable | From, To | Role |
| “Process invoice payments”, “3” | Process name, Process number | Property |

METAMODELING LANGUAGE 413

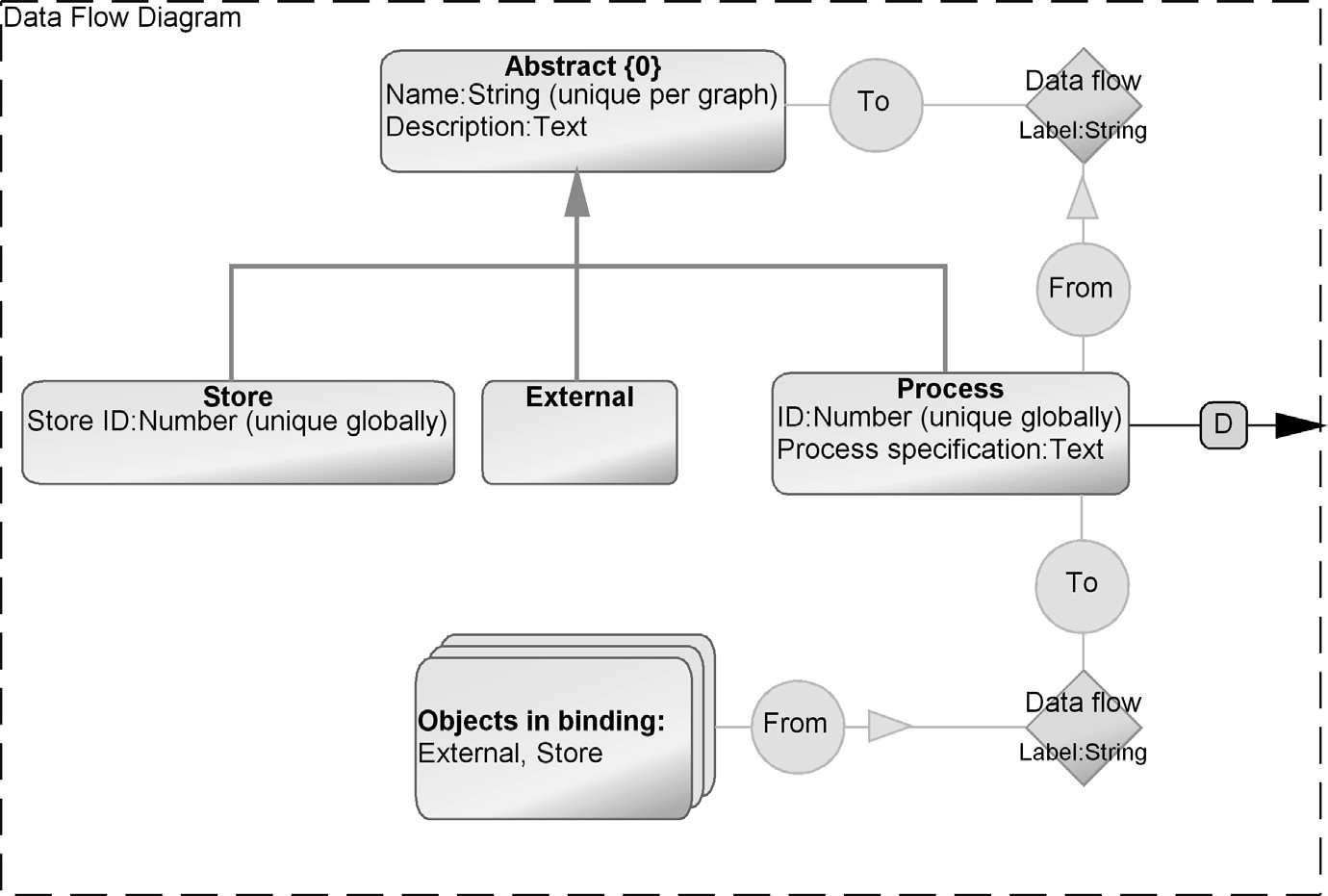


FIGURE A.2 A metamodel of data flow diagrams

We will call these basic constructs metatypes. Table A.1 lists some example instance elements of this model, their types, and their metatypes.

A language usually has additional constructs that are not immediately visible from one example model. In our example, a process may contain subprocesses that are specified in another data flow diagram. A language also has a number of constraints that ensure model correctness and consistency, such as:

. Processes can be connected with data flows to processes, externals, and stores . Stores and externals can be connected to processes, but not to stores or externals

. Each data flow connects two objects, with one from role and one to role

. Each process must have a process number, unique over all diagrams

. All processes in a diagram must have different process names

A metamodeling language can describe these constraints along with the modeling constructs. Figure A.2 illustrates the metamodel for data flow diagrams.

The metamodel is specified with GOPRR, a domain-specific language for specifying modeling languages. From a metamodel like this, modeling tool support can be created automatically without additional development tasks. Table A.2 shows the concepts and notation of GOPRR; as a shorthand, the names omit the word “type”: an “Object” here is actually an object type.

METAMODELING LANGUAGE

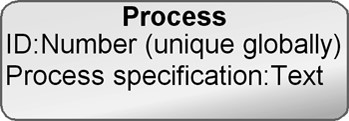
TABLE A.2 GOPRR Metamodeling Concepts

Metamodeling concept Notation

Graph specifies one modeling language, such as data flow diagram and use case diagram. A graph contains objects, relationships, and roles.

Object describes the basic concepts of a modeling language. Objects are the elements that you connect together and often reuse, such as process, state, and actor.

Relationship defines properties for the objects’ connections, such as data flow, inheritance, call, and transition. They are used to form bindings with objects and roles.



Role specifies the lines and end points of relationships, like the Subtype part of the inheritance relationship and the From part of the data flow relationship.

Property defines the attributes which characterize any of the previously mentioned language concepts. Properties can be of different data types (string, text, number, Boolean, collection, etc.) and link to other modeling language concepts or to external sources, such as files, programs, or web services. Examples Simple properties are defined inline in other concepts. Properties whose values are objects are shown like this:

of properties are process name, multiplicity,

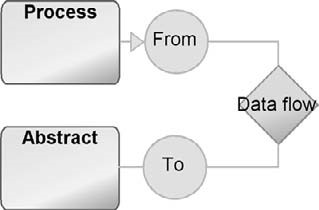


Binding

connects a relationship, two or more roles,

and for each role, one or more objects in a graph.

Binding is further specified with multiplicity.



Object Set

describes a collection of objects that can

play the same role in a binding, for instance,

External and Store can both be in the From role in

a data flow relationship.



Inheritance

allows creating subtypes of other language

concepts, for instance, External is a subtype of

Abstract.



Decomposition

allows objects to have subgraphs, for instance

a process can decompose to another data

flow diagram.



Explosion

allows objects, relationships, or roles to be

linked to other graphs, for instance, the detailed

structure of a Store in a data flow diagram may

be specified in an entity relationship diagram.



and data type.