UML Statechart Diagrams on the ADONIS Metamodelling Platform

Hans-Georg Fill

Department of Knowledge Engineering
University of Vienna
Vienna, Austria

Abstract

The metamodeling platform ADONIS has originally been implemented for the use in business process management. Its method independency and extensive customization functionalities also allow for the application in many different other areas such as strategic management, e-learning, object-oriented systems engineering, knowledge management, and numerous others. In computer science the Unified Modeling Language (UML) is the dominating standard for describing systems and behaviours. In this article it is shown how the abstract and concrete syntax of UML statechart diagrams can be described by the use of the metamodeling concepts of ADONIS.

Keywords: statechart diagrams, metamodeling, business process management, UML

1 Introduction

The original motivation for the research and development that led to the implementation of the ADONIS metamodeling platform as it exists today can be described as follows: the support of different frameworks which enable the modeling of current and anticipated future business application needs, the integration of the existing and the new information technology enterprise environment, and the provision of a continuous performance method for assessment and improvement of the running business [7]. Around ten years ago
when the research for ADONIS started at the University of Vienna most of the software tools that existed for business process management at this time had the disadvantage of being restricted to specific, not-modifiable methods and components [5]. The idea behind ADONIS was to develop a software toolkit that provided method-independence and could be adapted to the individual needs of a customer i.e. the user.

Following these guidelines ADONIS was implemented by BOC, a spin-off company of the University of Vienna, based on the concepts of metamodeling and has since then been very successfully deployed in the field of business process management, process-oriented application development, and numerous other application areas such as for example strategic management, e-learning or knowledge management [5,8,9,13]).

1.1 Related Work

The basic relationships of the syntax and semantics of modeling languages that are also relevant for this paper have been described besides others by Harel and Rumpe [2] who also put a special emphasis on UML. The Meta Object Facility and the Model Driven Architecture specifications of the OMG [20,19] can be regarded as an official attempt to consolidate the different approaches in metamodeling and will be both very relevant in the future. As the development of ADONIS already took place at a time when these specifications were still in a very early stage they have not been used for the past versions of the product - nevertheless as the need for standardization grows they might be integrated in future versions. A similar commercial product that uses metamodels for defining modeling languages is the Metacase MetaEdit+ toolkit [14] that also provides support for UML models. Further commercial toolkits for UML modeling that could be compared to ADONIS would be Borland Together [22], IBM Rational Software [4], Gentleware Poseidon for UML [1] or Microsoft Visio [15].

This paper is organized as follows: chapter 2 explains the characteristics of modeling methods as referred to in this paper, chapter 3 describes the modeling hierarchy and architecture of ADONIS and chapter 4 outlines the implementation of statechart diagrams in ADONIS with special emphasis on the abstract and concrete syntax.

2 Modeling methods

Today several types of modeling methods exist that are used in the area of business and information technology. They are either shaped to a specific field of application such as for example models for strategic management (e.g.
business process and organisational management (e.g. [5,11]) or workflow models (e.g. [3,6,12]) or can be used for object-oriented systems engineering like the models of the UML [16]. According to Karagiannis et al. [10] modeling methods consist of two components:

(i) A Modeling Technique and
(ii) Mechanisms and Algorithms

Modeling techniques can be further divided into a modeling language and a modeling procedure. The modeling language contains the elements for generating models and is described by its syntax, semantics, and notation (e.g. textual or graphical). The modeling procedure explains how the modeling language is applied to generate instances of models [10].

To describe the modeling language one approach is to define a metamodel as a model of the modeling language [20]. Thereby a hierarchy of modeling languages is built where a (meta-)modeling language in one layer describes the modeling language in the underlying layer [21]. The hierarchy of these modeling languages is theoretically not limited to a certain level nevertheless a useful level of abstraction has to be found [10]. Illustration 1 depicts these relationships graphically where the bottom level is the original that symbolizes the target point for all modeling ventures.

3 ADONIS Modeling Hierarchy and Architecture

The ADONIS metamodeling platform is based on a modeling hierarchy (see Figure 2) that is one possible instance of the theoretical considerations explained above. The ADONIS modeling hierarchy consists of three layers [5]:

(i) the ADONIS meta-meta (meta²) model,
(ii) the ADONIS metamodel and
(iii) a concrete model instance of the metamodel.

The meta-metamodel is implemented in C++. Modifications of the meta- metamodel can only be executed by ADONIS developers. By instanciating the meta-metamodel ADONIS metamodels (e.g., the basic metamodel for Business Process Management) are created that are described in the language ALL (ADONIS Library Language). From these metamodels further individual metamodels can be derived. Figure 3 shows an excerpt of the ADONIS meta-metamodel.

For the creation of individual metamodels in ADONIS a user can specify classes and relation classes as well as their attributes thereby defining the syntax of the modeling language. The semantics of the modeling language can either be specified in textual notation by adding attributes to the relevant classes and relation classes or by additional mechanisms and algorithms that can be coded in AdoScript, the scripting language of ADONIS, and also included in the ALL descriptions. The standard configuration of ADONIS contains a model editor, a simulation component for simulating business processes, and an analysis and evaluation component for querying and evaluating objects, relationships and their attributes.

The graphical notation of ADONIS is graph-based, the visualization of the classes (i.e., the nodes) and the relation classes (i.e., the edges) can be defined by a specific graphical representation grammar (GraphRep) that also allows for dynamic state changes based on attribute values. The instances of the metamodels (i.e., the models) are either described in the ADONIS definition language ADL or in XML.
The software architecture of ADONIS is realized as a three-tier-architecture consisting of the database facilities and ADONIS core components, the logical layer containing the application components (e.g. modeling, analysis, simulation and evaluation components) and the user-interface layer [5].

4 UML Statechart diagrams in ADONIS

In the following it shall be referred to the ADONIS UML library version 2.1 together with ADONIS version 3.6 that contains metamodels allowing for the representation of activity, class, collaboration, component, deployment, package, sequence, statechart, and use case diagrams of UML [16].

4.1 Abstract syntax

In its current implementation the modeltype State Diagram of the ADONIS UML library contains seven modeling elements (start, state, end, selfdelegation, note, 'has successor' and 'has note') to represent the basic entities of the proposed structure of the UML notation guide for statechart diagrams [16, pp.3-136 – 3-150]. The selfdelegation class is a specificity for the representation in ADONIS to allow for the modeling of transitions that have their start and end point directed to the same state. The State Diagram modeltype is part of the ADONIS UML library which contains a user-defined class hierarchy based on the ADONIS metamodel.

The modeling element State shall be used as an example to show how
modeling elements are defined (the complete listing of the ALL source code is omitted here due to the limitations of space): The ADONIS metamodel defines a class modelElement that is derived from a pre-defined ADONIS metaclass (GP-Konstrukt$^3$) and provides a set of attributes and default values for creating modeling elements. From modelElement a user-defined class is derived named stateElement that serves as a meta-class for the classes representing the statechart elements e.g. the State modeling element.

The modeltype State Diagram contains two relationclasses has successor and has note. The classes and the relationclasses are interconnected by ‘Is From Class’ and ‘Is To Class’ relationships that define which relationclass connects which classes respectively meta-classes.

Both the classes and the relationclasses can have sets of attributes that can be further constrained by facets. Classattributes signify attributes that contain common values for all instances of the referenced class or relationclass.

In Example 4.1 the ALL code is shown that corresponds to the definition of the modelElement class, the stateElement class, and the State class as well as the definition of the has successor relationclass. The classes from the ADONIS metamodel are symbolized by double underscores.

**Example 4.1** ALL excerpt for the syntax definition of state diagram classes:

```
//...
CLASS <modelElement> : <_GP-Konstrukt_>  
//...
CLASS <stateElement> : <modelElement>  
//...
CLASS <State> : <stateElement>  
//...
RELATIONCLASS <has successor>  
    FROM <stateElement>  
    TO <stateElement>
```

Within the State class several attributes are defined (additional to the attributes inherited from the meta-classes) that are relevant for the specific behaviour of the class. Among them are classattributes for the description of the graphical representation (GraphRep-attribute), the enumeration of attributes that can be modified by the user in the modeling environment (Attr-Rep-attribute) or the restriction of cardinalities to other objects. Example 4.2 shows the ALL syntax definition for the two classattributes GraphRep and Klassenkardinalität(class cardinality).

In comparison to currently discussed proposals respectively already exist-

$^3$ This metaclass is part of the ADONIS business process metamodel that has also been used as the basic metamodel for the UML library.
ing specifications for the standardization of the abstract and concrete syntax of UML models [17,18] ADONIS integrates the definition of the graphical representation into the metamodel description thereby allowing for the possibility to inherently define the graphical representation of each object as well as to link properties of the graphical representation directly to classes and attributes of the metamodel. This leads to advantages especially for the display and editing of models in the ADONIS model editor that requires additional information to correctly display and edit the models (e.g. about the layout or the context of the respective model).

Example 4.2 Excerpt of the ALL code definition of classattributes:

```plaintext
//...
CLASSATTRIBUTE <GraphRep>
VALUE "
  GRAPHERP
  SHADOW off FILL color:black
  ELLIPSE rx:0.15cm ry:0.15cm AVAL d:'Darstellung'
  IF (d = "mit Namen")
    ATTR "Name" y:-.2cm w:c h:b
  ENDIF"
//...
CLASSATTRIBUTE <Klassenkardinalität>
VALUE "CARDINALITIES
  RELATION "has successor"
  max-outgoing:1
  max-incoming:0"
//...
```

4.2 Concrete syntax

The concrete syntax for instances of statechart diagrams in ADONIS can be described either in the ADONIS Definition Language ADL or in XML based on the ADONIS Document Type Definition for XML models. ADONIS provides both import and export components for these languages. The next release of ADONIS (3.8) that was not yet available at the time of writing this paper will include XML Metadata Interchange (XMI) functionalities for UML that allows for the exchange of UML models with other tools based on the UML metadata profiles of the OMG.

In example 4.3 an excerpt of the ADL Syntax is shown. The TYPE element specifies the model type and includes additional attributes for describing the model specific information such as the author of the model, the date and time of creation or settings for the visual representation. The INSTANCE elements define concrete instances of the modeling elements specified in the metamodel with attributes specific to the ADONIS modeling environment (e.g.
the position of the according graphical object). With the RELATION element instances of the relation classes of the metamodel are defined including the instances of the 'Is From Class' and 'Is To Class' relationships. Figure 4 shows the ADONIS modeling environment with a model instance of a statechart diagram, the notebook component that allows the editing of the attributes of the class instances, and an excerpt of the ADONIS administration toolkit for the editing of the class hierarchy.

![Fig. 4. ADONIS modeling and administration environment](image)

**Example 4.3** Excerpt of the ADL code for concrete model instances of statechart diagrams:

```adl
//...
TYPE <State Diagram>
  ATTRIBUTE <Autor>
  VALUE "testuser"
//...;
INSTANCE <Active> : <state>
  ATTRIBUTE <Position>
  VALUE "NODE x:11.49cm y:7.98cm w:4.82cm h:2.54cm index:1"
//...
RELATION <has successor>
  FROM <Active> : <state>
  TO <Idle> : <state>
//...
5 Conclusion and outlook

In this paper it was outlined how an implementation of UML statechart diagrams can be accomplished on the ADONIS metamodeling platform. For this purpose ADONIS provides different languages for the definition of the abstract and concrete syntax of arbitrary metamodels. With the most recent release of ADONIS it will also be possible to exchange UML models in XML Metadata Interchange (XMI) format thereby allowing users to import UML models in other modeling and CASE tools that also support this standard.

In the current implementation of statechart diagrams in ADONIS semantic properties can only be described in textual notation. Further work would have to be done to integrate operational semantics for statechart diagrams in ADONIS. One possibility to achieve this is through the scripting language AdoScript that can be used to evaluate the modelled objects and relationships and also generate new models based on predefined rules.

References


