19 Papyrus: A UML2 Tool for Domain-Specific Language Modeling

Sébastien Gérard[1], Cédric Dumoulin[2], Patrick Tessier[1], and Bran Selic[3]

[1] CEA LIST, Laboratory of Model Driven Engineering for Embedded Systems (LISE), Boîte courrier 65, Gif sur Yvette Cedex, F-91191 France
{Sebastien.Gerard,Patrick.Tessier}@cea.fr
[2] LIFL and INRIA-Lille Nord Europe, University of Lille, France
Cedric.Dumoulin@lifl.fr
[3] Malina Software Corp., Nepean, Ontario, Canada
Selic@acm.org

Abstract. This chapter outlines Papyrus, a tool for graphical modeling of UML2 applications. It is an open-source project, designed as an Eclipse component, and based on the existing EMF-based realization of the UML2 meta-model. The goal of this open-source project is twofold. First, it is a complete, efficient, robust, and methodologically agnostic implementation of a UML2 tool to both industry and academia. Second, it is an open and flexible facility for defining and utilizing domain-specific modeling languages using a very advanced implementation of the UML profile concept.

Keywords: UML2, UML profile, DSL, MDD, MDE, Modeling and Eclipse.

19.1 Introduction

As part of its Model-Driven Architecture (MDA) initiative, the Object Management Group (OMG, http://www.omg.org) – an international consortium representing numerous industrial and academic institutions – has provided a comprehensive series of standardized technology recommendations in support of model-based development of both software and systems in general. These cover core facilities such as meta-modeling, model transformations, and general-purpose and domain-specific modeling languages. A key component in the latter category is UML (the Unified Modeling Language, [1]), which has emerged as the most widely used modeling language in both industry and academia.

A number of tools supporting UML are available from a variety of sources. However, these are generally proprietary solutions whose capabilities and market availability are controlled by their respective vendors. This can be problematic for industrial users, who may require highly-specific tool capabilities as well as long-term support which, from a vendor perspective, often extend beyond the point of commercial viability. Consequently, some industrial enterprises are seeking open-source solutions for their UML tools. Of course, the fact that open-source solutions are often less costly is another influencing factor, although it is typically not the primary motivation for selecting them, since there is always a cost involved in integrating such solutions. Indeed, industrial developers using open-source solutions need time and money to account for debugging or developing new features in their projects.

A similar situation is encountered in research which typically depends on open source tools, as most proprietary products are too constraining and inflexible to allow optimized implementation of new ideas and prototypes.

Therefore, the Eclipse platform along with its Model Development Tools (MDT, `http://www.eclipse.org/modeling/mdt`) subproject is the environment of choice for developing open-source tools for modeling. Indeed, MDT focuses on big "M" modeling within the Modeling project of Eclipse. To achieve this goal, MDT aims at providing: (a) an implementation of industry standard meta-models, and (b) exemplary tools for developing models based on those meta-models. The UML2 meta-model implementation was its first component (also known as the "UML2 Component", `http://wiki.eclipse.org/MDT-UML2`). This component has become the *de facto* standard implementation of the UML2 meta-model (note that it is also the basis for the UML2 tool suites provided by IBM). Based on the UML2 component, several open-source tools emerged providing facilities for graphical modeling within Eclipse. In early 2008, three of the main community initiatives developing such a tool —MOSKitt (`http://www.moskitt.org/eng/`), Papyrus (`http://www.papyrusuml.org`) and TOPCASED (`http://www.topcased.org`) — decided to merge their efforts and provide a joint contribution to the Eclipse MDT. This new graphical editor, named as Papyrus (`http://wiki.eclipse.org/MDT/Papyrus-Proposal`), was accepted by the Eclipse's Project Management Committee in August 2008, and the first code delivered in November 2008: `http://wiki.eclipse.org/MDT/Papyrus`.

The following section outlines the architecture of the Papyrus tool and its main capabilities in terms of UML2 graphical modeling and openness for customization. Section 19.3 provides some snapshots of UML2 diagrams created with Papyrus illustrating its graphical capabilities. Finally, section 19.4 gives some conclusions and a roadmap for the tool.

19.2 Capabilities

As explained previously, Papyrus is a graphical editing tool for UML2. It is Eclipse-based and it uses the Eclipse Graphical Modeling Framework (GMF, http://www.eclipse.org/modeling/gmf/). This section first provides an overview of Papyrus capabilities and its architecture, and then expands on two of its distinguishing features: Subsection 19.2.3, describes the general Papyrus capabilities for graphical UML2 modeling, while subsection 19.2.4 focuses on the abilities of the tool to be customized for domain-specific needs through the use of UML profiles to define domain-specific modeling languages (DSLs).

19.2.1 Overview

Papyrus is a tool consisting of several editors, mainly graphical editors but also completed with other editors such as textual-based and tree-based editors. All these editors allow simultaneous viewing of multiple diagrams of a given UML model. Modifying an element in one of the diagrams is immediately reflected in others diagrams showing this element. Papyrus is integrated in Eclipse as a single editor linked to one UML 2 model. Papyrus provides a main view, showing model diagrams, and additional views including an outline view, a property view, and a bird's-eye view. The multiple diagrams are managed by Papyrus rather than by Eclipse. Model diagrams can be arranged in tabbed views, and several tabbed views can be arranged side by side (left, right, top and bottom). Such views can be created, re-sized, moved, and deleted by dragging them to the desired position.

Papyrus is highly customizable and allows adding new diagram types developed using any Eclipse-compatible technology (GEF, GMF, EMF Tree Editors, ...). This is achieved through a diagram plug-in mechanism. In fact, even the default diagrams use this mechanism, allowing their easy replacement if desired.

19.2.2 Global Architecture and Design Tenets

As shown in Fig. 19.1, the Papyrus top-level architecture consists of two main parts: a core providing the common services used by the various diagram editors, and a part containing the various model editors.

One of the main functions of the core component of Papyrus is to enable collaboration of the different editors regardless of their specific implementation technology (e.g., EMF or GMF). The main services provided by the papyrus core are:

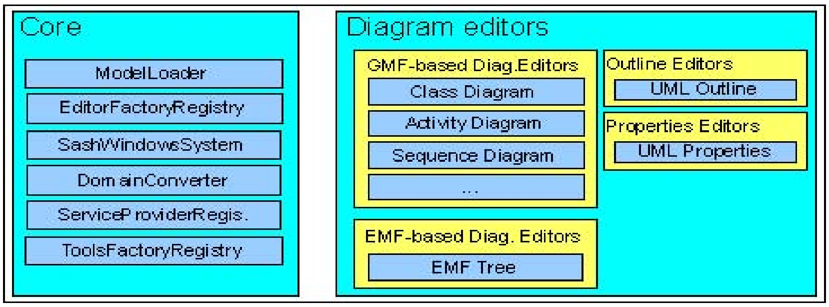


Fig. 19.1. This figure sketches the architecture model of the papyrus tool

- *A sash windows system.* This provides the ability to have multiple diagrams opened simultaneously, and to arrange them as desired.
- *Model life-cycle support.* This allows loading and saving of models.
- *Pluggable diagram editor factories.* A diagram factory is used to create a particular kind of diagram in Papyrus. For example, there is a factory for

class diagrams. The plug-in mechanism allows registering such factory in the core, enabling thereby the management of existing diagrams via the sash window system.
- *Pluggable tools factories.* A tool factory allows creation of a tool that will interact with the model. Examples of such tools are: code generators, transformations engines, model checkers, and refactoring facilities.

The Papyrus core is not linked to a particular technology other than EMF, allowing use of various different technologies. The core also provides facilities that enables diagrams and tools using the same implementation technology (e.g., GMF) to share classes.

In Papyrus, the UML meta-model and the graphical models are at the heart of the proposed architecture: diagram editors and other external tools can be used to define and modify any models, but they can also be used to observe them and react accordingly. Thus, diagram editors and external tools can be independent while still reacting to each other's actions.

Both the property view and the outline editors are also independent model editors in Papyrus. This means that they can interact with the model like any other diagram editors, providing hence another interesting and useful point of view on the model.

Several diagram editors have been developed and integrated successfully in Papyrus, using different technologies like GEF, GMF, EMF Tree Editors, SWT.

The UML part of Papyrus consists of diagrams partly generated with GMF, and of a property view editor also generated with a dedicated framework. Section 19.3 provides some snapshots of UML diagrams edited with Papyrus.

19.2.3 UML2 Graphical Modeling Capabilities

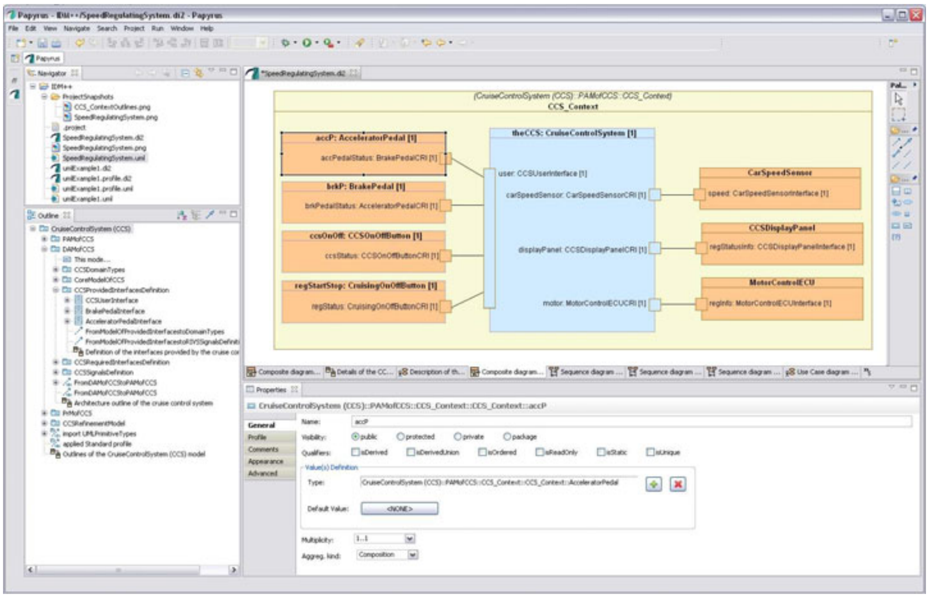
Papyrus allows supporting all the diagram types defined in UML2. Hence, a Papyrus user can build UML2 models that strictly conform to the standardized specification of the language. Papyrus can then be used to check against the UML specification: if something cannot be represented in Papyrus, this generally means that it is not allowed by the specification.

Papyrus provides a lot of functions in support of UML 2 modeling. The following list is not exhaustive (due to space limitation), but is still a representative sample of the advanced modeling features of Papyrus:

- *A tools palette.* Each diagram type has its own palette of tools allowing creation of UML elements in the diagram.
- *A property view.* This allows the editing of any property of a UML element as well as any of its graphical properties. These properties are organized by categories for ease of use. There is also an 'expert' category, allowing access to all the properties defined in the UML meta-model.
- *Contextual accelerator actions.* When the cursor is moved to an area where some actions are available, these actions are shown in a popup near the cursor. An action can then be selected directly without going to the palette.

– *Contextual text editors enabling syntax highlight, completion and content assist.* When a text-based model element is accessed, e.g., a property, Papyrus helps by providing a list of possible values, such as a list of existing types, or existing cardinalities. Papyrus also validates the text according to its grammar (if the latter is defined). This functionality is customizable: one can provide one`s own implementation (e.g., a specific text editor and validation rules) for specific needs.
– *OCL constraint specification and checking.* OCL constraints can be specified for each UML element at the model or at the meta-model level, such as in a profile definition. Furthermore, these constraints can be checked against the current model.
– *Model import.* Papyrus can import models, profiles or elements from other files. Imported elements can be used in the model and edited the same as any other elements.

Figure 19.2 illustrates the default perspective of the Papyrus tool: The diagram editor is in the top right of the figure, with the corresponding palette to its right. At the bottom right is the properties view editor. This editor is used to edit, and modify (if required), the properties of the element selected in the diagram editor or in the project outline editor. The latter is located in the bottom left part of the figure. Finally, in the top left part of the figure is the file system navigator, which enables access to different existing projects.



Papyrus - IDM++/SpeedRegulatingSystem.di2 - Papyrus
File Edit View Navigate Search Project Run Window Help
Navigator
Outline
*SpeedRegulatingSystem.di2
(CruiseControlSystem (CCS)::PAMofCCS::CCS_Context)
CCS_Context
accP: AcceleratorPedal [1]
accPedalStatus: BrakePedalCRI [1]
brkP: BrakePedal [1]
brkPedalStatus: AcceleratorPedalCRI [1]
ccsOnOff: CCSOnOffButton [1]
ccsStatus: CCSOnOffButtonCRI [1]
regStartStop: CruisingOnOffButton [1]
regStatus: CruisingOnOffButtonCRI [1]
theCCS: CruiseControlSystem [1]
user: CCSUserInterface [1]
carSpeedSensor: CarSpeedSensorCRI [1]
displayPanel: CCSDisplayPanelCRI [1]
motor: MotorControlECUCRI [1]
CarSpeedSensor
speed: CarSpeedSensorInterface [1]
CCSDisplayPanel
regStatusInfo: CCSDisplayPanelInterface [1]
MotorControlECU
regInfo: MotorControlECUInterface [1]
Composite diagram...
Details of the CC...
Description of th...
Composite diagram...
Sequence diagram ...
Sequence diagram ...
Sequence diagram ...
Use Case diagram ...
Properties
General
Profile
Comments
Appearance
Advanced

CruiseControlSystem (CCS)::PAMofCCS::CCS_Context::CCS_Context::accP

Name: accP
Visibility: [✓ public] protected private package
Qualifiers: isDerived isDerivedUnion isOrdered isReadOnly isStatic isUnique
Value(s) Definition
Type: CruiseControlSystem (CCS)::PAMofCCS::CCS_Context::CCS_Context::AcceleratorPedal
Default Value: <NONE>
Multiplicity: 1..1
Aggreg. kind: Composition

Fig. 19.2. Default perspective of Papyrus

19.2.4 Building DSL Tools Profiling the UML2

In accordance with its primary goal of realizing the UML2 specification, Papyrus provides extensive support for UML profiles. It includes all the facilities for defining and applying UML profiles in a very rich and efficient manner. It also provides powerful tool customization capabilities similar to DSML-like (Domain Specific Modeling Language) meta-tools. The main intent here is to enable the profile plug-in of Papyrus to dynamically drive its customization as defined by a profile. This means that when applying a profile, the tool may adapt its visual rendering and GUI to fit the specific domain supported by that profile. Of course, if the profile is unapplied later, the tool resumes to its previous configuration.

When designing a UML2 profile, it may be necessary to customize one or more existing UML2 diagram editors. For that purpose, Papyrus supports customization of existing editors with the added capability of extending such customizations by adding new tools relevant to the stereotypes defined in the UML profile. For example, the SysML requirements diagram editor is designed as a customization of the classical UML2 class diagram editor with additional features for direct manipulation of all the concepts defined in the SysML requirements diagram (see example show in section 19.3).

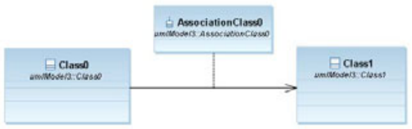
Finally, when embedding a profile within an Eclipse plug-in, a designer can also provide a specific properties view that will simplify (i.e., make more user friendly) the manipulation of the stereotypes and their related properties. The outline editor and the menu of the tools can also be customized to fit domain-specific concerns appropriate to the profile.

19.3 Case Study

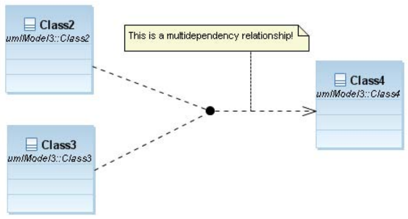
The purpose of this section is to provide the reader some sense of the facilities of Papyrus in terms of UML2 graphical modeling. It is, of course, infeasible to give the full set of possibilities offered by the tool in a short summary article such as this. Hence, this section will only illustrate one specific papyrus editor—the class diagram editor.

The following two figures represent a class association between two classes and a multipoint-dependency relationship linking three classes (two sources and one target) respectively.

The Class diagram example is very illustrative of the Papyrus tenets in terms of its implementation goal: 99,9% of the specification without imposing any methodological constraints or assumptions. This is very important because it is common in many tools that some particular form of UML2 usage is not supported because of hard-coded implementation choices. The intent of basic Papyrus is to not favour any specific methodology and thereby offer an unhampered access to the full extent of UML2. Furthermore, it is intended to give methodologists and users the ability to construct specific customizations of the tool to support methodologies that are best suited to given profiles. That is, when applied to a



(a) Example of class association modeling



(b) Example of multipoint dependency relationship modeling

user model, such a profile also reconfigures the GUI and behavior of the Papyrus tool to fit a domain-specific methodology.

The following figure illustrates the palette customization feature of Papyrus. This palette provides both standard UML tool capabilities (called "UML Links" and "UML Elements" respectively) as well as an extended set of domain-specific tool capabilities required for the EAST_ADL2.0 language profile. (EAST-ADL2.0 is an architectural description language, intended for modeling automotive embedded systems (`http://www.atesst.org/`)).

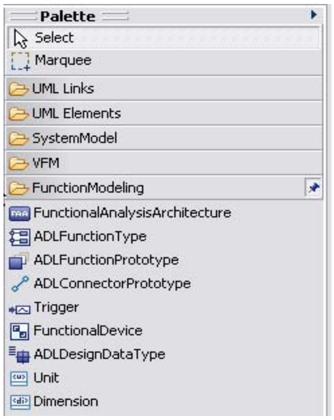


Fig. 19.3. Example of customized palette

19.4 Conclusions and Future Work

This article introduces the Papyrus tool, an Eclipse-based graphical editor for UML2 modeling. This open-source application was devised with two principal objectives in mind. First, it aims at implementing the complete UML specification (currently in its 2.2 revision), enabling thereby its potential use as the reference implementation of the OMG standard. Furthermore, it was designed as a highly scalable and robust tool for supporting large-scale industrial projects (as indicated by the growing list of industrial supporters). The second principal objective of Papyrus was to provide an open and highly customizable tool for defining domain-specific languages and corresponding tools via the UML profile mechanism. This facilitates interchange with other tools supporting the UML standard and also takes advantage of the widespread acceptance of UML 2. Consequently, Papyrus provides an efficient and effective alternative to custom and proprietary DSL tools, without losing the benefits of an international standard.

At present, a significantly revised version of Papyrus is under development with a major delivery milestone set for mid-year 2009. This version will support all 13 diagram types of UML2 and will also provide the fullest support for profiles of any UML tool currently available, whether open-source, or commercial. The next major deliverable is set for mid-year 2010, when Papyrus will be integrated into the principal Eclipse release schedule whereby key components are packaged and released jointly each year.

Finally,it is intended that Papyrus serve as an experimental platform for researchers who need to construct proof of concept prototypes. Built on top of Eclipse as an open-source project, Papyrus is an ideal candidate for this purpose.

Reference

[1] OMG: UML Version v2.1.2, `http://www.omg.org/spec/UML/2.1.2/`