Papyrus

Dominique Heer

University of Antwerp

Abstract

This paper evaluates Papyrus, a UML2 tool for domain-specific language modeling. By using an example from the image processing domain, Papyrus' abilities to serve as a graphical modeling environment are examined. In detail, UMLs extension mechanisms are exploited to implement a domain-specific modeling language for creating simple image processing pipelines. In addition, Papyrus is compared to mbeddr, an extensible set of languages based on C.

Keywords:
Papyrus, UML, UML Profiles, mbeddr, Image Processing, DSL, MDE

1. Introduction

This section gives a brief overview of model-driven engineering and the problems and challenges that lead to the development of an open-source graphical modeling environment.

1.1. Overview

It should be no surprise that more and more companies integrate model-driven engineering (MDE) in their software development life cycle, for it has a number of advantages when compared to other methodologies which focus on the computing aspects and not on domain models. For example, MDE captures domain knowledge on a high level of abstraction, and the resulting models can serve as a documentation. In addition, it is less error-prone and validation becomes easier because it is executed on the model and not on code-level.

Email address: dominique.heer@student.uantwerpen.be ()

As a logical consequence, many modeling languages exist that visualize the design of a system, with UML (Unified Modeling Language) being the most prevalent one. Standardized by the Open Management Group (OMG), UML2 features fourteen different diagram types. However, it has been found that for some applications UML is too general, and further specification and adaption is needed. To support this, UML profiles (see 2.4) have been added. Profiles allow users to extend the meta-model of UML, thus creating new elements suitable for specific domains.

Furthermore, many tools exist that aim to support the MDE development process. However, in the last 25 years, it turned out that not a single company was able to build a MDE tool which captures all requirements needed by their customers. In addition, researchers work with their own tools, while companies use proprietary platforms. This leads to the fact that technology and knowledge exchange becomes unnecessarily difficult. At last, most commercial tools do not offer sufficient support for DSMLs (Bordeleau (2014)).

To tackle the problems described above, CEA, Ericsson and other companies have worked together to build an open source modeling tool called Papyrus which is based on the Eclipse platform. Although the main focus of Papyrus is UML modeling, it can be customized in various ways (see section 2). In this paper, Papyrus' abilities to serve as a graphical modeling environment for a specific domain is explored and evaluated. In detail, a DSL (domain-specific language) for image processing pipelines is implemented (see 3 and 4). In section 5, the experiences made are then compared to another project which focused on implementing the same DSL in another tool, mbeddr[1]

1.2. Related Work

The idea to implement a domain-specific language for image processing pipelines is not new: in Hartmann et al. (2015), IPOL, a DSL for image processing applications, is described. IPOL is a textual language based on XML. It allows the user to specify the complete pipeline, that is, images sources (sensors), processing blocks, and a target display (sinks). By abstracting the development of a pipeline, the design becomes language- and hardware-independent and thus can be easier analyzed and maintained.

[1] `http://mbeddr.com`

Furthermore, Membarth et al. (2016) implemented HIPA^{cc}, a DSL embedded in C++ for describing image processing algorithms. They found that developers oftentimes have difficulties choosing the right programming language, especially when it comes to parallel computation, and therefore a DSL for modeling image processing algorithms would be a compelling solution. In addition, they implemented a source-to-source compiler which translates from the DSL to the target architecture.

Profiles in UML have also been broadly discussed. Fuentes-Fernández and Vallecillo-Moreno (2004) give a short and informative introduction to UML Profiles with special focus on the extension mechanisms and how to use them. Accordingly, Atkinson and Kühne (2002) describe some flaws of the UML extension mechanism and how to overcome them.

2. Papyrus

In the following sections, the Papyrus tool is introduced.

2.1. Overview

Within the Eclipse project, a variety of tools exist that address the development of domain-specific languages, such as Sirius[2], Xtext[3] and, in a broader sense, Papyrus. Whereas Sirius allows to easily create graphical modeling workbenches, and Xtext supports the development of text-based DSLs, Papyrus on the other hand is primarily used for UML2 modeling, since it implements 100% of the OMG specification. However, by making use of UML profiles (see 2.4), it can be extended and diagrams can be customized. In addition, Papyrus supports code generation from UML diagrams 2.5.

2.2. History

As described in section 1.1, most commercial modeling tools did not meet all the requirements of the users. Therefore, CEA List, a French research institute, started to develop Papyrus, but soon other companies showed interest in the collaborative development of a modeling tool. In 2008, a proposal was submitted to the Eclipse foundation with the goal to implement Papyrus on

[2] `http://www.eclipse.org/sirius`
[3] `http://www.eclipse.org/Xtext`

top of the Eclipse framework. According to the proposal[4], this was a natural choice because Eclipse is open source, offers powerful frameworks (EMF, GMF, see 2.3) and has a vibrant user community.

In October 2016, Papyrus version 2.0.1 was released.

2.3. Architecture

Papyrus is a sub-project of Eclipse's Model Development Tools (MDT) based on the Graphical Modeling Framework (GMF) and the UML2 metamodel. The latter is in turn based on EMF (Eclipse Modeling Framework), a framework which allows to model a data model and generate code from it. EMF models can be constrained by OCL, the Object Constraint Language. GMF on the other hand provides components for developing graphical modeling editors based on EMF.

Figure 1 summarizes the information given above.

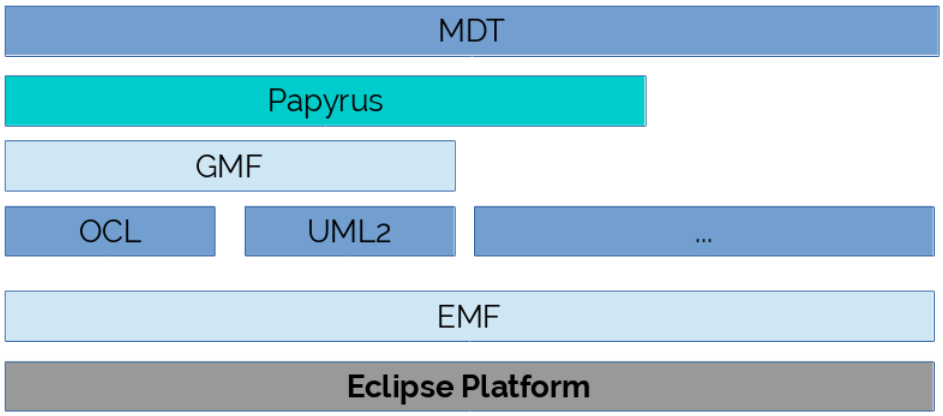

Figure 1: The Papyrus architecture

2.4. UML Profiling

To uphold it's reputation as a flexible graphical modeling environment, Papyrus has to allow the user to create custom DSLs for specific domains. In some domains, it is not necessarily useful to use the UML standard, but

[4] `http://wiki.eclipse.org/MDT/Papyrus-Proposal`

rather a specialized language. For these cases, UML features extension mechanisms to customize and extend the standard diagrams. These mechanisms are defined as follows:

1. stereotypes: allow the user to extend UMLs vocabulary by altering the meaning, syntax and visual representation of UML model elements
2. constraints: impose restrictions on the metamodel
3. tagged values: further enrich stereotypes by adding attributes

These three mechanisms describe a so-called UML profile. Papyrus has extensive support for UML profiles. For example, the SysML modeling language, which is based on UML and focuses on systems engineering, was implemented within Papyrus by using UML profiles.

Fuentes-Fernández and Vallecillo-Moreno (2004) provides a more detailed overview and example-based introduction to UML profiles.

In the scope of this project, profiles will be used to add new diagram types to support the image processing pipeline DSL (see 3).

2.5. Code Generation

Despite its ability to edit UML and SysML diagrams, Papyrus also allows the user to automatically generate code. As of writing this, Java and C++ are integrated, with support for Ada and C planned. In addition, custom code generators can be added. These generators are developed in the Xtend[5] programming language, a Java dialect which aims to improve Java by adding both new functionality (macros, powerful switch statements, operator overloading, etc) and by removing boiler plate. Xtend compiles into Java code and can therefore be used together with any Java library. Furthermore, it seamlessly integrates into the Eclipse IDE.

The ultimate goal of this project is to implement a model-to-code generator for the image processing pipeline DSL.

3. Case study

3.1. Description

For the case study, a simple image processing pipeline was chosen. Such a pipeline consists of one or more input images, several processing blocks,

[5] `http://www.eclipse.org/xtend`

and one or more output images. Processing blocks can be for example image scaling, noise reduction, filters, effects or colorspace conversion. Technically, such a pipeline is an acyclic directed graph, with nodes representing the components (source images, processing blocks, target images), and edges being the data flow.

Figure 2 shows an example for a graphical representation of an image processing pipeline. This pipeline features two input images, two processing blocks and one output image.

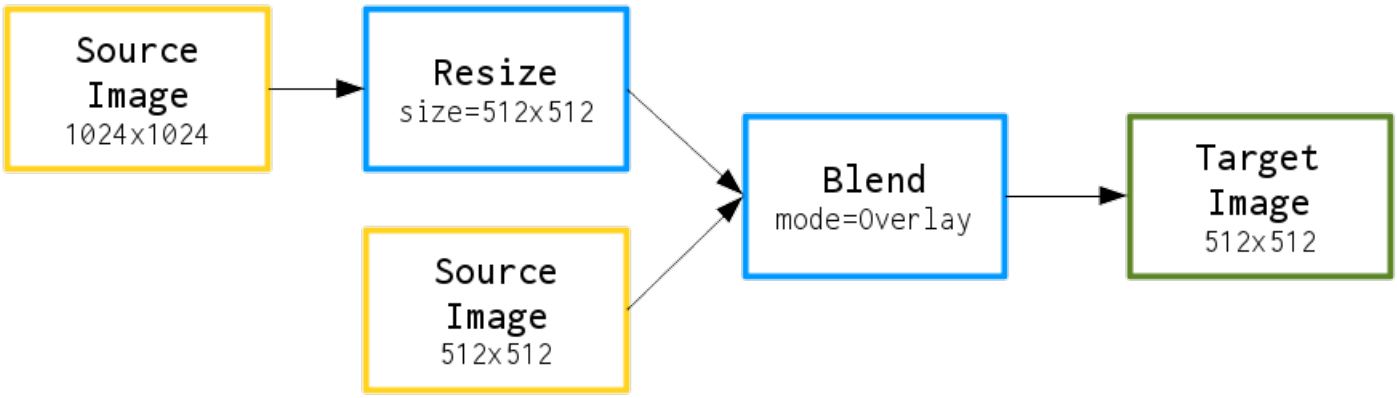

Figure 2: A simple example for a image processing pipeline

Purpose of this case study is to serve as an example for implementing a DSL in Papyrus. In addition, the same DSL is implemented in another project with the tool embeddr. Both tools, embeddr and Papyrus, will be compared in section 5 based on the experiences made.

3.2. Language Design

4. Implementation in Papyrus

5. Comparison with embeddr

6. Conclusion

7. References

Atkinson, C., Kühne, T., Jul. 2002. Profiles in a strict metamodeling framework. Sci. Comput. Program. 44 (1), 5–22.

Bordeleau, F., 2014. Model-based engineering: A new era based on papyrus and open source tooling. In: OSS4MDE@MoDELS.

Fuentes-Fernández, L., Vallecillo-Moreno, A., 2004. An introduction to uml profiles.

Hartmann, C., Reichenbach, M., Fey, D., 2015. Ipol - a domain specific language for image processing applications. Proceedings of the International Symposium on International Conference on Systems (ICONS 2015), 40–43.

Membarth, R., Reiche, O., Hannig, F., Teich, J., K"orner, M., Eckert, W., 2016. Hipacc: A domain-specific language and compiler for image processing. IEEE Transactions on Parallel and Distributed Systems 27 (1), 210–224.