Migrating from proprietary tools to open-source software for EAST-ADL metamodel generation and evolution

Citation for the original published paper (version of record):
Holtmann, J., Steghöfer, J., Lönn, H. (2022). Migrating from proprietary tools to open-source software for EAST-ADL metamodel generation and evolution. Proceedings - ACM/IEEE 25th International Conference on Model Driven Engineering Languages and Systems, MODELS 2022: Companion Proceedings: 7-11. http://dx.doi.org/10.1145/3550356.3559084

Migrating from Proprietary Tools to Open-source Software for EAST-ADL Metamodel Generation and Evolution

Jörg Holtmann
jorg.holtmann@cse.gu.se
Chalmers | University of Gothenburg
Gothenburg, Sweden

Jan-Philipp Steghöfer
jan-philipp.steghofer@cse.gu.se
Chalmers | University of Gothenburg
Gothenburg, Sweden

Henrik Lönn
Henrik.Lonn@volvo.com
Volvo Technology AB
Gothenburg, Sweden

ABSTRACT

Open-source software has numerous advantages over proprietary commercial-off-the-shelf (COTS) software. However, there are modeling languages, tool chains, and tool frameworks that are developed and maintained in an open-source manner but still incorporate COTS tools. Such an incorporation of COTS tools into an overall open-source approach completely annihilates the actual open-source advantages and goals. In this tool paper, we demonstrate how we eliminated a COTS tool from the otherwise open-source-based generation and evolution workflow of the domain-specific modeling language East-Adl, used in the automotive industry to describe a variety of interdisciplinary aspects of vehicle systems. By switching to a pure open-source solution, East-Adl becomes easier to inspect, evolve, and develop a community around. We compare both the mixed COTS/open-source and the open-source-only workflows, outline the advantages of the open-source-only solution, and show that we achieve equivalent tooling features compared to the original approach.

CCS CONCEPTS

• **Software and its engineering → Domain specific languages**; **Architecture description languages**; **System modeling languages**; **Development frameworks and environments**; **Software development techniques**.

KEYWORDS

Model-based engineering, DSL construction, open-source

ACM Reference Format:
Jörg Holtmann, Jan-Philipp Steghöfer, and Henrik Lönn. 2022. Migrating from Proprietary Tools to Open-source Software for EAST-ADL Metamodel Generation and Evolution. In *ACM/IEEE 25th International Conference on Model Driven Engineering Languages and Systems (MODELS '22 Companion), October 23–28, 2022, Montreal, QC, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3550356.3559084

1 INTRODUCTION

Software-intensive, automotive systems continue to increase in complexity (see, e.g., [1]). Model-based engineering (MBE) provides one way to cope with this complexity. Domain-specific modeling languages such as East-Adl (*Electronics Architecture and Software Technology—Architecture Description Language*) [6, 8] provide suitable concepts and abstractions for automotive engineers. East-Adl covers aspects of requirements, verification & validation, product lines, functional, software, and hardware architectures, and software allocation for the interdisciplinary engineering lifecycle phases. The language evolves to address new and changing needs under the auspices of an industry consortium [8].

The modeling tool suite Eatop [10] provides the tooling capabilities for East-Adl and is applied in a prototyping environment (together with further language derivatives) at Volvo Technology AB. Eatop is based on the Eclipse Modeling Framework (EMF) [12] and the Eclipse Sphinx project [13], and it provides a number of modeling and convenience features such as validators as well as a custom serialization format.

Developing Eatop is itself a complex endeavor. The research project Maenad [22] has addressed this complexity with an approach that distinguishes between a metamodel for documentation purposes (*documentation MM*) and an implementation metamodel (*implementation MM*) that is deployed to Eatop. Particularly, the approach generates the structurally different implementation MM as well as other artifacts, such as customized model and edit code plugins for EMF, from the documentation MM. This generation of the Eatop modeling infrastructure decouples the model and edit code plugins specific to a concrete East-Adl version from the remaining Eatop plugins which provide the actual tooling features. This means that East-Adl can evolve separately from Eatop. Umanovskis and Voget [27] report that this way of working has been adopted from Artop [2], the official open-source editor for the automotive software modeling language and architecture framework Autosar [3], which is closely aligned with East-Adl.

However, this approach relies on the proprietary commercial-off-the-shelf (COTS) tool Sparx Enterprise Architect [26] for the modeling of the language. Using this commercial and proprietary software has the drawback that it prevents potentially interested stakeholders from producing new versions of the language and experimenting with the tool chain. It also leads to vendor lock-in. Furthermore, Sparx Enterprise Architect itself and particularly the interface to access its models relies on the COTS operating system Microsoft Windows, thereby preventing applying the generation approach on other platforms and requiring users to buy further COTS software. Since East-Adl and Eatop are both developed as open-source projects and are intended to be community-driven, these restrictions have been a major hurdle in development and in evolving the language to fit the needs of automotive engineers in different organizations. The commercial, proprietary, and platform-specific basis of the approach thus impedes the application and further involvement of a broader community.

MODELS '22 Companion, October 23–28, 2022, Montreal, QC, Canada

ACM ISBN 978-1-4503-9467-3/22/10.
https://doi.org/10.1145/3550356.3559084

In contrast, open-source software (OSS) addresses these needs by providing the freedoms to use, distribute, inspect, and modify the software system [28]. Consequently, Carillo and Okoli [5] report on many positive follow-up effects of OSS on the society, like the establishment of control over and trust in the software that is used. At the same time, OSS — and in our case in particular the Eclipse Modeling Framework (EMF), Ecore, and the associated modeling tools — have reached a maturity that makes it feasible to model and evolve even complex languages like East-Adl in an open-source environment. We therefore present the migration from an approach that uses COTS components to an Eclipse-native approach for generating the Eatop modeling infrastructure. The approach uses an EMF-based documentation metamodel as the single source of truth and as the basis for the model and edit code.

The main advantage of this new approach is that it supports rapid prototyping, because there is no tool gap and powerful Eclipse-based technologies are now at the language engineers disposal. At the same time, it avoids vendor lock-in, is platform-independent, and provides better maintainability. We also show that we can achieve the same documentation features as the COTS tool with EMF and argue how we enable the possibility for (meta-)model co-evolution.

In the upcoming Section 2, we provide background information about the workflow and the technical realization of the mixed COTS/open-source realization. Subsequently, we describe in Section 3 our pure open-source solution and compare it to the original one. Finally, we conclude and outline future work in Section 4.

2 BACKGROUND: ORIGINAL APPROACH FOR GENERATING THE EATOP MODELING INFRASTRUCTURE

Both the interdisciplinary systems modeling language East-Adl and the software modeling language and software architecture framework Autosar emerged out of the research project East-Eea [15]. Thus, both languages are closely aligned with each other to ensure a seamless model-based development process. This language alignment was designed on the meta-metamodel level M3, on the metamodel/language level M2, and on the tooling level.

On the meta-metamodel level M3 for the definition of both languages, the basic notions of the MetaObject Facility (MOF) [20] are applied and supplemented by Autosar rules [4]. On the metamodel or language level M2, East-Adl and Autosar share several abstract metaclasses, and East-Adl references certain concepts in the Autosar metamodel. On the tooling level, East-Adl and Autosar share the evolution and maintenance of their respective documentation MM's in Sparx Enterprise Architect as well as a transformation approach from these documentation MM's to Open-source and Eclipse-based tooling platforms. That is, like for East-Adl and Eatop, there is a transformation approach [27] to generate from the Autosar documentation MM an implementation MM for the Autosar Tooling Platform (Artop) [2]. Thus, the transformation from the East-Adl documentation MM to the Eatop implementation MM was designed likewise [27].

In the following, we describe this transformation approach on a conceptual level (Section 2.1) as well as on a technical realization level (Section 2.2).

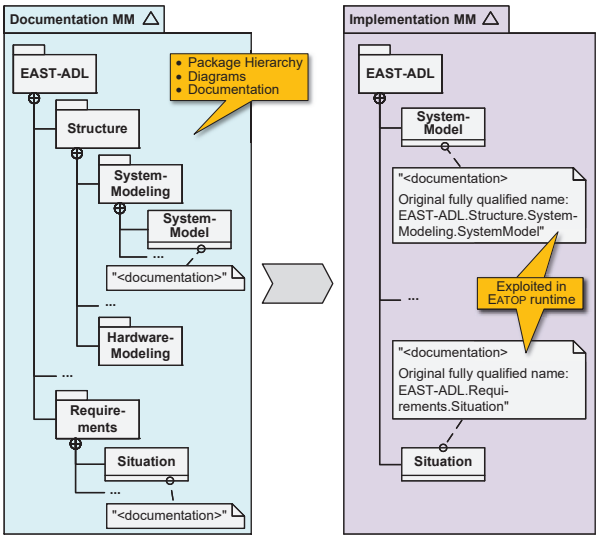

Figure 1: General Approach for Transforming an East-Adl Documentation MM to an Eatop Implementation MM

2.1 Distinction and Transformation Between Documentation and Implementation MM

As outlined in the introduction, the original approach distinguished between an East-Adl documentation MM as well as a structurally different Eatop implementation MM and realized a transformation approach to generate the latter one from the former one. Figure 1 depicts this general transformation approach with the East-Adl Documentation MM on the left-hand side and the Eatop Implementation MM on the right-hand side.

The Documentation MM has the purpose of aligning the MM in its specific version with the corresponding version of the East-Adl specification (cf. [7]). This encompasses structuring the MM into packages that correspond to the specification structure (e.g., the metaclass SystemModel in the package Structure.SystemModeling and the metaclass Situation in the package Requirements), providing diagrams for each package, and providing documentation texts for the particular metaclasses (cf. annotations "<documentation>")— where the diagrams and documentation texts also show up in the specification.

The Implementation MM is the basis for generating model and edit code which applies a customized serialization format. This follows the standard approach used by the Eclipse Modelling Framework (EMF) to generate a number of plugins which are deployed into the generic Eatop framework. To this end, the transformation approach flattens the package hierarchy from the Documentation MM. However, the original hierarchy is persisted by extending the original metaclass annotations with information about it. That is, all metaclasses (e.g., SystemModel and Situation) are rearranged directly under the root package EAST-ADL, and their documentation annotations are extended by their original fully qualified names (cf. annotations "<documentation> Original fully qualified name: ..."). Both the actual documentation and the original fully qualified name in these annotations are exploited at runtime in Eatop.

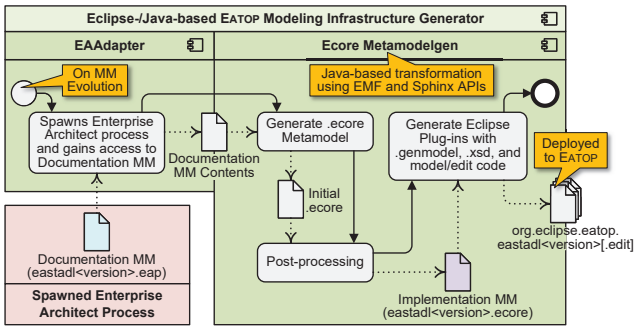

Figure 2: Original Workflow for the MM Transformation with the Commercial and Proprietary Tool Enterprise Architect

2.2 Transformation based on Sparx Enterprise Architect

In this section, we describe the original transformation approach described in Section 2.1. Whereas the code for this transformation is published as OSS [9], it relies on the COTS tool Sparx Enterprise Architect for maintaining the versions of the documentation MM and is therefore not fully supported by OSS.

Figure 2 depicts the internal, technical workflow of the original realization. The main focus of the figure is on the Eclipse-/Java-based Eatop Modeling Infrastructure Generator with its two subcomponents EAAdapter and Ecore Metamodelgen. Additionally, the Spawned Enterprise Architect Process simply provides access to the Documentation MM maintained in Sparx Enterprise Architect. This Documentation MM is stored in a file with the naming schema eastadl<version>.eap, where .eap is the default name suffix for Enterprise Architect model files.

When the workflow is executed (typically after the metamodel was evolved), the component EAAdapter spawns a process of Sparx Enterprise Architect and, using this tool's API, gains access to the Documentation MM. In this process, the EAAdapter reads the Documentation MM Contents and transfers them to an in-memory representation based on Ecore. This representation can then be read by the component Ecore Metamodelgen. This component is realized by means of Java and uses the APIs of both EMF and Sphinx. From the Documenation MM Contents, the component generates an Initial .ecore file with the same structure than the original documentation MM. In the subsequent Post-processing step, particularly the transformation to the Implementation MM is conducted as described in Section 2.1. Furthermore, many technical aspects specific to Sparx Enterprise Architect models are transformed to EMF-specific concepts, and the metamodel is prepared according to the expectations of the Eatop framework. For these purposes, the transformation approach applies a multitude of post-processing templates. In the final step, the component Ecore Metamodelgen generates Eclipse plugins including a generator model, an EMF XMI schema, and the model/edit code from the Implementation MM. These plugins, which are specific to an East-Adl version, are then deployed to Eatop.

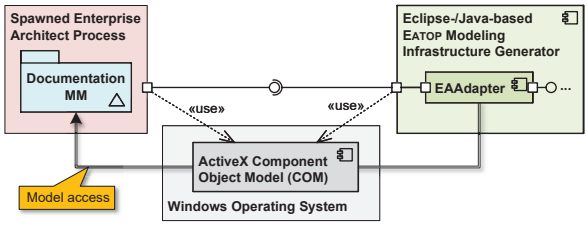

Figure 3: Technical, Platform-dependent Principle for Accessing Enterprise Architect Artifacts from Java

Figure 3 depicts the technical principle behind gaining access to the Documentation MM maintained in Sparx Enterprise Architect. This tool's API relies on the ActiveX Component Object Model (COM) technology provided by the COTS operating system Microsoft Windows. In order to execute the EAAdapter component and thereby gain access to the Documentation MM, Sparx Enterprise Architect as well as Windows have to be installed on the host computer, and the corresponding .dll library file has to be copied to the Java implementation. Beyond the general disadvantages of commercial and proprietary software mentioned in the introduction, the need for this platform-dependent and awkward setup is a particular issue in the application of the original approach.

3 ECLIPSE-NATIVE APPROACH FOR GENERATING THE EATOP MODELING INFRASTRUCTURE

In the upcoming Section 3.1, we describe the technical realization of our pure open-source approach for the generation and evolution workflow for East-Adl. Subsequently, we compare in Section 3.2 the open-source-only approach with the mixed COTS/open-source one and discuss the advantages of the pure open-source approach.

3.1 Realization based on Eclipse

Figure 4 depicts the internal, technical workflow of our new Eclipse-native generator for the general transformation approach described in Section 2.1. We again realize the component in Java and apply the EMF and Sphinx APIs. Similarly to the original workflow, the workflow is started due to the evolution of the East-Adl metamodel and takes as input a documentation MM that is structured as explained in Section 2.1. In contrast to the original workflow, this documentation MM is stored as an .ecore file and has been created and maintained using the documentation and diagramming features of EcoreTools [14] included in EMF. In the step Post-processing (which is realized by means of an adapted subset of the original post-processing templates), the Documentation MM gets transformed into the Implementation MM. From this Implementation MM, the Eclipse plugins including a generator model, an EMF XMI schema, and the model/edit code are generated, like in the original approach.

In the documentation MM, the textual documentation for the particular metaclasses and the package-wise diagrammatic documentation is conducted by means of the EMF-native component

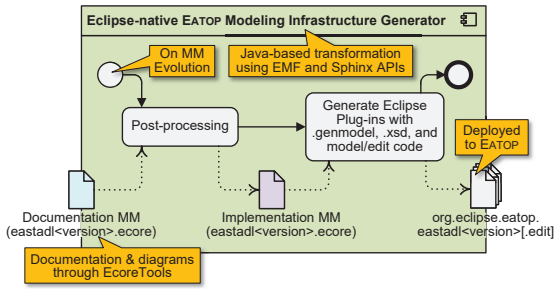

Figure 4: New Eclipse-native Workflow for the MM Transformation

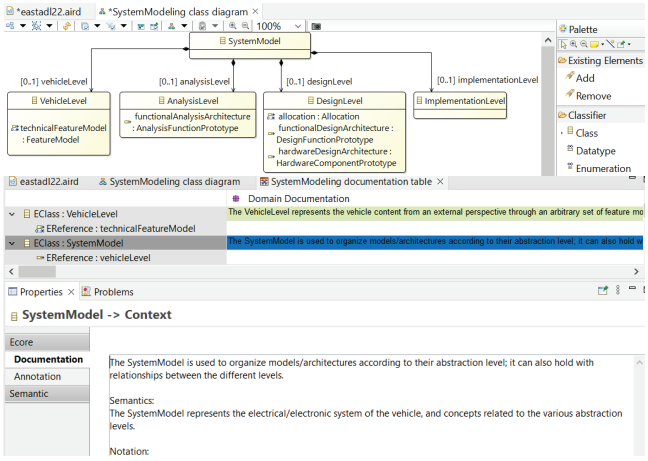

Figure 5: Screenshots—Package-wise Class Diagrams and Class-wise Documentation through EcoreTools

EcoreTools. Figure 5 depicts screenshots of this approach. The upper part of the figure shows a class diagram for the package EAST-ADL.Structure.SystemModeling (cf. left part of Figure 1). Like Enterprise Architect and most modeling tools, the EcoreTools diagram editor provides a model element palette, the actual diagram canvas with typical layouting capabilities, and a property editor. For the same package, the middle part of the figure depicts a documentation table, which provides a row-wise list of the package's metaclasses that can be selected for editing (i.e., metaclass SystemModel in the figure). In contrast to Sparx Enterprise Architect where the user has to open a property editor for each class, the documentation table provides a convenient overview of the documentation texts for all classes of a package. The bottom part of the figure shows the text editor for the selected table row. This text editor provides the same text editing capabilities than the property editor of Sparx Enterprise Architect.

We provide the new approach including a comprehensive user documentation in [18].

3.2 Improvements in the New Approach

Our proposed approach offers **equivalent documentation features** to the original one. The diagramming and layouting features of EcoreTools are essentially the same as in Sparx Enterprise Architect. Regarding the textual documentation of the metaclasses, we favor the EcoreTools documentation approach with a package-wise table overview over the documentation approach of Sparx Enterprise Architect. Thus, we consider the features of the EcoreTools documentation approach for maintaining the documentation MM at least equivalent to the features provided by the COTS tool.

The direct comparison of Figure 4 with Figure 2 also shows that our Eclipse-native approach is simpler and more straightforward than the original approach. This also implies a **better maintainability** of the transformation approach. As Klint et al. [19] have shown, the maintainability of language implementations increases when using dedicated tools for the development of external domain-specific languages. Likewise, Goldschmidt and Kübler [17] cite Oman and Hagemeister [21] in the context of DSLs who emphasize that maintainability depends — amongst other things — on the management practices and the software environment, both of which become simpler with our new, Eclipse-native approach.

Furthermore, as our Eclipse-native, **fully open-source** approach does not require buying a COTS tool and a COTS operating system anymore, the software we make available [18] enables all interested stakeholders to try, reenact, reuse, and improve the approach as well as East-Adl itself. Enabling these basic tenets of OSS [28] will hopefully have positive long-term effects on the development of East-Adl since the foundations of the language become available to a broader audience, including potential adopters that do not have a license of Sparx Enterprise Architect, but also researchers and students, traditionally strong contributors to East-Adl.

At the same time, our migration from the original approach relying on a COTS tool to OSS **avoids vendor lock-in** for Eatop stakeholders. Although Sparx Enterprise Architect has been actively developed for over 20 years now [25], it is still a risk to fully rely on only one tool vendor. EMF is under active development as almost as long as Sparx Enterprise Architect [16] and is available as open-source itself. This means that even though projects of the Eclipse Foundation can also get archived, it is possible to invest own resources to keep technology alive.

The Eclipse-native approach also means that maintaining the documentation MM and generating the implementation MM as well as the Eatop modeling infrastructure is now **platform-independent**. It relies only on Eclipse and Java and has been tested on Microsoft Windows, macOS, and Linux.

Most importantly, however, the Eclipse-native approach improves **rapid prototyping**. Since the documentation MM and the transformation to the implementation MM are maintained and executed in the same tool, more convenient rapid prototyping of changes in the metamodel can be achieved. Furthermore, the East-Adl language is modular in the sense that new language concepts can be easily hooked into the language infrastructure by setting the concepts' super-types to abstract metaclasses. Together with the new feature that no export or use of bridge technologies are necessary, a change in the metamodel can be directly tested with new model and edit code.

At the same time, the powerful **(meta-)model co-evolution** mechanisms like Eclipse Edapt™ [11] or one of its extensions (see, e.g., [29]) could be used with our Eclipse-native approach, which is missing in the original approach. Eclipse Edapt™ works directly

on Ecore metamodels and models. That means that the history of changes in the metamodel can be directly used to create migrators for existing models. The original approach would not be amenable to such a workflow since the documentation MM was maintained in Sparx Enterprise Architect and a change history is therefore not available to Eclipse Edapt™.

Rapid prototyping and (meta-)model co-evolution can have a major impact on the future development of East-Adl. The benefits of rapidly prototyping languages has been shown in different domains, for example, for domain-specific languages for wireless sensor networks [24]. Rouvoy and Merle [23] show that the ability to rapidly prototype architecture description languages makes it easier to develop new, useful architectural designs. The ability to quickly make changes to the East-Adl specification and see the impact on the editors and how these changes are represented in the tooling directly will make it easier for the East-Adl Association to explore new directions for the language. At the same time, the ability to use Eclipse-native tools to create migrators for new language versions will allow the users of East-Adl to easily and seamlessly adopt new versions of the language when they become available. This ability also supports rapid prototyping as the impact of language changes on instances can also be inspected directly.

4 CONCLUSION AND FUTURE WORK

In this tool paper, we show how we eliminated a COTS tool from the otherwise open-source-based generation and evolution workflow of East-Adl. We describe the functional and technical principles behind both approaches, compare them, and discuss the advantages of the pure open-source solution.

By eliminating the COTS tool from the workflow, we establish a pure open-source solution and enable all open-source advantages, which were earlier annihilated by the incorporation of the COTS tool. The advantages encompass rapid prototyping, the avoidance of vendor lock-in, platform-independency, a better maintainability, equivalent documentation features, as well as the possibility of (meta-)model co-evolution.

In terms of future work, the version 2.2 of East-Adl is currently in the beta draft [7], so that the final version will be released in near time. Thus, we plan to apply Eclipse Edapt™ [11] or one of its extensions (e.g., [29]) to evaluate the (meta-)model co-evolution w.r.t. the former East-Adl version 2.1.12 and thereby set the basis for a state-of-the-art evolution management for future language versions. Furthermore, due to the close alignment of East-Adl with Autosar (cf. Section 2), one could create a similar open-source-only solution for the generation and evolution workflow of the Artop environment.

ACKNOWLEDGMENTS

Parts of this research were sponsored by Vinnova under grant agreement nr. 2019-02382 as part of the ITEA4 project BUMBLE.

REFERENCES

[1] Vard Antinyan. 2020. Revealing the complexity of automotive software. In *28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1525–1528. https://doi.org/10.1145/3368089.3417038

[2] Artop User Group. 2022. *Autosar Tool Platform (Artop)*. Retrieved August 2022 from https://www.artop.org/

[3] Autosar Consortium. 2022. *Automotive Open System Architecture (Autosar) Website*. Retrieved August 2022 from https://www.autosar.org/

[4] Hans Blom, Henrik Lönn, Frank Hagl, Yiannis Papadopoulos, Mark-Oliver Reiser, Carl-Johan Sjöstedt, De-Jiu Chen, and Ramin Tavakoli Kolagari. 2013. *East-Adl—An Architecture Description Language for Automotive Software-Intensive Systems*. White Paper, Version 2.1.12. Retrieved August 2022 from http://maenad.eu/public/conceptpresentations/EAST-ADL_WhitePaper_M2.1.12.pdf

[5] Kevin Carillo and Chitu Okoli. 2008. The Open Source Movement: A Revolution in Software Development. *Journal of Computer Information Systems* 49, 2 (2008), 1–9.

[6] East-Adl Association. 2022. *East-Adl Bitbucket Repository*. Retrieved August 2022 from https://bitbucket.org/east-adl/

[7] East-Adl Association. 2022. *East-Adl Specifications*. Retrieved August 2022 from https://www.east-adl.info/Specification.html

[8] East-Adl Association. 2022. *East-Adl Website*. Retrieved August 2022 from https://www.east-adl.info

[9] East-Adl Association. 2022. *Eatop Metamodel Generator*. Retrieved August 2022 from https://bitbucket.org/east-adl/east-adl/src/Revison/org.eclipse.eatop/metamodelgen/

[10] East-Adl Association. 2022. *Eatop Resources*. Retrieved August 2022 from https://bitbucket.org/east-adl/east-adl/src/Revison/org.eclipse.eatop/

[11] Eclipse Foundation. 2022. *Eclipse Edapt™*. Retrieved August 2022 from https://www.eclipse.org/edapt/

[12] Eclipse Foundation. 2022. *Eclipse Modeling Framework (EMF)*. Retrieved August 2022 from https://www.eclipse.org/modeling/emf/

[13] Eclipse Foundation. 2022. *Eclipse Sphinx*. Retrieved August 2022 from https://www.eclipse.org/sphinx/

[14] Eclipse Foundation. 2022. *EcoreTools*. Retrieved August 2022 from https://www.eclipse.org/ecoretools/

[15] Electronics Architecture and Software Technology—Embedded Electronic Architecture (East-Eea) Project Consortium. 2018. *East-Eea impact story: Paving the way towards revolutionary automotive software development*. Retrieved August 2022 from https://itea4.org/project/result/download/7181/EAST-EEA%20impact%20story.pdf

[16] Eclipse Foundation. 2003. *EMF 1.x Downloads*. Retrieved August 2022 from https://www.eclipse.org/modeling/emf/downloads/dl-emf1x.html

[17] Thomas Goldschmidt and Jens Kuebler. 2008. Towards Evaluating Maintainability Within Model-Driven Environments. *Software Engineering 2008*.

[18] Jörg Holtmann. 2022. *Source code and user documentation for the Eclipse-native Eatop modeling infrastructure generator*. Retrieved August 2022 from https://bitbucket.org/east-adl/east-adl/src/Revison/org.eclipse.eatop/genmodelcodegen/

[19] Paul Klint, Tijs Van Der Storm, and Jurgen Vinju. 2010. On the impact of DSL tools on the maintainability of language implementations. In *Tenth Workshop on Language Descriptions, Tools and Applications*. 1–9. https://doi.org/10.1145/1868281.1868291

[20] Object Management Group (OMG). 2022. *MetaObject Facility*. Retrieved August 2022 from https://www.omg.org/mof/

[21] Paul Oman and Jack Hagemeister. 1992. Metrics for assessing a software system's maintainability. In *Conference on Software Maintenance 1992*. IEEE, 337–338. https://doi.org/10.1109/ICSM.1992.242525

[22] Maenad Project Consortium. 2014. *Maenad—Model-based Analysis & Engineering of Novel Architectures for Dependable Electric Vehicles*. Retrieved August 2022 from http://www.maenad.eu/

[23] Romain Rouvoy and Philippe Merle. 2012. Rapid prototyping of domain-specific architecture languages. In *15th ACM SIGSOFT Symposium on Component Based Software Engineering*. 13–22. https://doi.org/10.1145/2304736.2304741

[24] Daniel A Sadilek. 2007. Prototyping domain-specific languages for wireless sensor networks. In *4th Int. Workshop on Software Language Engineering*. Citeseer, 76–91.

[25] SparxSystems Ltd. 2022. *About Sparx Systems*. Retrieved August 2022 from https://sparxsystems.com/about.html

[26] SparxSystems Ltd. 2022. *Sparx Enterprise Architect*. Retrieved August 2022 from https://www.sparxsystems.eu/enterprise-architect/enterprise-architect-editions

[27] Daniels Umanovskis and Stefan Voget. 2014. *Eatop: An East-Adl Tool Platform for Eclipse*. Technical Report Deliverable D5.3.1. Model-based Analysis & Engineering of Novel Architectures for Dependable Electric Vehicles (Maenad Project). Retrieved August 2022 from http://www.maenad.eu/public/Deliverables/MAENAD_Deliverable_D5.3.1_EATOP_V3.0.pdf

[28] Jovica Đurković, Vuk Vuković, and Lazar Raković. 2008. Open source approach in software development—Advantages and disadvantages. *Management Information Systems* 3 (2008), 29–33.

[29] Y Vissers, JGM Mengerink, Ramon RH Schiffelers, Alexander Serebrenik, and Michel A Reniers. 2016. Maintenance of specification models in industry using Edapt. In *2016 Forum on Specification and Design Languages (FDL)*. IEEE, 1–6. https://doi.org/10.1109/FDL.2016.7880374